# RESEARCH STATEMENT

Pooyan Jamshidi (pjamshid@cs.cmu.edu)

## Summary

It has been generally recognized that one-size-fits-all software systems are neither effective nor efficient in many domains; hence, software systems have increasingly become more configurable. However, the ability to configure systems causes complexity, as the number of configurations grows exponentially with the number of options. Although configurability appears attractive, developers are overwhelmed with the difficulty of understanding important properties (e.g., performance, energy consumption) of configurable systems. This trend is unsettling: as society's dependence on software increases, our overall understanding of these systems is diminishing.

My goal is to understand the **performance** behavior of **highly-configurable software** systems especially in **dynamic** and **uncertain** environments and exploit this understanding to build dependable and reliable software systems. Characterizing performance behavior has been traditionally associated with prohibitively high cost. My research challenges the accepted trade-off between accuracy and cost, aiming to simultaneously improve performance prediction and decrease the cost of building performance models. My agenda is to exploit the growing power of machine learning to enable building of reliable performance models using cheap sources that provide an approximation of the performance of the system. To realize this agenda, I have:

1. Laid out the foundations of a theory based on empirical investigations about how similar the performance behavior is across environments (answering why and when we can build cheap performance models) [1].

2. Developed methods and tools to build reliable performance models using transfer learning [2, 3].

3. Employed performance models in feedback loops [4] to enable software systems to perform as desired (high performance and low energy) in uncertain environments (*e.g.*, workload suddenly changes) [5, 6].

4. Employed performance models to guide the refactoring of software architecture [7], and created architectural principles to realize performance-aware software systems in the cloud [8].

I am a software engineering researcher, but in my goal of enabling software systems with desired performance behavior, I incorporate *statistical machine learning, control theory*, and *distributed systems*.

**Societal impact**: With an abundance of cloud providers, small and medium-sized companies are developing configurable software with cloud services. However, the cost of failure for the software-enabled businesses are high (*e.g.*, the failed launch of the Obamacare website (healthcare.gov), or Google services outages [9]) and, therefore, the systems need to be reliable [10]. My goal is to enable building reliable and dependable software by enabling developers to understand the performance influence of configurations and to make tradeoff.

To make an impact, my approach is to choose relevant problems, build practical solutions, and apply the solutions to real-world systems. I therefore interact with developers from companies and open-source projects, while taking their feedback into account. For instance, I collaborated closely with **Intel** during my PhD; our collaboration led to an auto-scaling product that is integrated with OpenStack [5, 11]. I have also developed auto-scaling controllers for Microsoft Azure [12], and studied the performance behavior of real-world systems: CoBot (robotics) [2], SaC (compiler) [1], x264 (encoder) [1], Hadoop [3], Apache Storm/Spark [3], deep neural networks, and diverse cloud systems [5, 6, 12]. I have also developed an configuration optimization tool for big data systems [13] that was shortlisted for the Imperial college venture capital in 2016.

## Vision: Enabling performance tradeoff for highly-configurable systems

My research is driven by observations about how modern software typically fails to meet its goals in dynamic and uncertain environments [5], and how such failures arise from a lack of understanding of performance behavior [9, 10]. The main challenge is that software systems are becoming increasingly complex. As a result, developers and users are not able to fully comprehend the performance behavior of software systems. The complexity arises from the demand for tailor-made configurable software instead of one-size-fits-all solutions. Configurability introduces complexity: A software system with $n$ independent binary configuration options, results in $2^n$ different ways of

configuring the system. With billions of potential configurations, one cannot gain understanding of key system properties, because it is unfeasible to measure the performance of every configuration independently.

The goal of my research is to conquer the lack of performance understanding of highly-configurable software, both in terms of improving prediction and decreasing the cost of building performance models. My research enables users/developers to reason about qualities (energy usage vs speed) and make tradeoff (sacrifice speed to save energy). This research is important and practical, but at the same time, it is an extremely difficult problem to solve: multi-objective optimization that requires a search in high-dimensional configuration spaces. In the presence of configurability, complexity of building performance models cannot be avoided entirely. However, much of the complexity of building performance models for configurable systems stems from our lack of understanding about the similarities of performance behavior across environments. **My core insight** is that, despite the high cost of measurement, learning performance models can become surprisingly cheaper, as long as we can reuse information across environments [1]. In my recent ASE paper [1], I demonstrated that there exists several sources of performance behavior that remain similar in different environments (e.g., under different workloads). My work shows that we can decrease the cost of learning performance models by exploiting such similarities using transfer learning. As an example, in the CMU BRASS MARS, a **DARPA** sponsored project on model-based adaptation of mobile robotics software, I have shown that we can considerably decrease the cost of learning power models for real robots via transfer learning by taking measurements from the simulators of the robots [2].

### Research approach

To realize my research goal, I apply a wide range of different methods, including instrumenting systems and building machine learning techniques and models. I insist on strong evaluations with real-world systems (e.g., configurable systems in my recent ASE paper [1]). To address the increasingly interdisciplinary nature of modern software-intensive systems, I collaborate not only with software engineers, but I also reach out to researchers in cyber-physical systems, performance engineering, and self-aware computing. I report issues and propose solutions to open-source projects, e.g., the Unity's ml-agents for developing intelligent systems (`https://goo.gl/9kgUGh`).

## 1 Current Research

### 1.1 Performance understanding of configurable software using transfer learning

**Motivation**. While today's software systems have given users more features and choices via configurable options, users need to understand the performance influence of options. To this end, a typical approach is to build a performance model of the systems via sensitivity analysis. Creating such models is challenging because of [2] : (i) *the curse of dimensionality*, and (ii) *the high costs of sampling*. Also, existing work assumes a *fixed environment*, should the environment change, a new model must be learned from scratch.

**My contribution**. My work addresses these challenges using transfer learning from other environments, in which the cost of sampling is typically cheaper compared to the target environment (*e.g.*, a simulator of a robot [2], or a lighter workload [1]). Similar to humans who learn from previous experience and transfer the learning to accomplish new tasks, I exploit knowledge about performance behavior gained in one environment to learn models for changed environments with **low cost**. My recent empirical study on real-world systems [1] uncovered that there exist different kinds of knowledge that are shared between environments. I have shown that this shared knowledge can potentially enable **performance testing, tuning, and optimization** of highly-configurable systems by concentrating on interesting regions.

My work addresses not only performance model learning, but also the application of performance models for (i) performance tuning: BO4CO (Bayesian Optimization for Configuration Optimization) [3], TL4CO (Transfer Learning for Configuration Optimization) [14], (ii) runtime computing resource adaptations: RobusT2Scale (Type-2 Fuzzy controller for Robust auto-Scaling) [12], FQL4KE (Fuzzy Q-Learning for Knowledge Evolution) [5], HMC (Hybrid Memory Controller) [6], FSL (Fuzzy SARSA Learning) [11], and (iii) performance-aware DevOps (performance feedback from operation to development). The self-learning cloud controllers [5] are able to learn optimal adaptation policies without any input from users by just interacting with the environment.

**Impact**. My recent work [1] on discovering performance similarities brings potential implications for other domains, such as model checking that suffer from scalability challenges. I am collaborating with my colleague

Javier Cámara to make probabilistic model checking scalable using transfer learning. Beyond publications, my work led to an auto-scaling product that is integrated with OpenStack through a collaboration with Intel.

### 1.2 Performance tuning of configurable software

**Motivation**. The performance difference between the best and worst configuration of a software system is typically several orders of magnitude [3], providing a case for finding the optimal configuration. The performance measurements of configurable software create a response surface that is strongly non-linear, non-convex, and multi-modal [3]. Therefore, finding the best configuration is a difficult problem.

**My contribution**. I developed BO4CO [3] to identify a near optimal configuration of highly-configurable software. BO4CO is a configuration optimization tool that enables software developers to find optimal configurations within limited experimental budgets. One of the application areas of BO4CO is to big data applications, which involve composing several configurable frameworks (*e.g.*, Apache Hadoop, Spark). This configuration space has high dimensionality and a naïve search algorithm will typically fail to find the optimal setting within limited time. Other challenges include measurement noise and local minima in the response surface. BO4CO performs Bayesian optimization using Gaussian Processes to model a response surface and marginal likelihood maximization as a learning mechanism. I have also developed TL4CO [14] that exploits past experimental data to accelerate configuration optimization for the most recent version of the system.

**Impact**. Besides publications, BO4CO and TL4CO solutions are in the process of being acquired by TATA.

## 2 Future Research

While pursuing my long-term goal of developing a theory that facilitates understanding performance behavior of configurable systems, I plan to develop techniques that allow (i) cyber-physical systems to self-repair, and (ii) developers to make informed decisions about configuration changes. This is important because systems are becoming increasingly more configurable, and we need to enable systems and developers to handle the complexity as a result of configurability by empowering them with the right techniques and tools.

### 2.1 Bio-inspired self-repair for cyber-physical systems (CPS)

**Motivation**. In recent years, CPSs have increased in number and complexity: unmanned vehicles, service robots, drones, planetary rovers. The software pieces embedded in these systems implement algorithms that enable them to accomplish their missions. These systems operate on limited resources, and most crucially, often work in close proximity to humans, and thus need to operate under stringent safety and reliability requirements. CPSs are also highly-configurable, and building such systems requires many design decisions for embedded software. These software-level configurations can impact performance, *e.g.*, poor configurations for a drone may cause the battery to run out while carrying out a mission. In addition, CPSs operate in highly dynamic and uncertain environments, prone to failure at runtime.

**My research plan**. In this research, I use my expertise in developing techniques for highly-configurable systems to enable CPS to adapt at runtime: (i) I will develop novel techniques to enable the configuration optimization (for energy saving purposes) of CPSs at runtime using statistical machine learning; (ii) I will develop methods that enable CPSs to **recover from failure at runtime** using nature inspired healing [15]; (iii) I will also investigate coordination protocols to coordinate the actions of self-optimization and self-repair. Inspired by self-healing in natural organisms, I will develop techniques to enable modular CPSs that are able to merge to form new structures, split into separate components with independent controllers, and self-heal by removing or replacing malfunctioning components. For realizing this, I rely on my expertise in machine learning to build techniques that are able to learn the dynamics of environments.

**Expected impact**. My vision is that, in the future, CPSs will no longer be designed and built for a specific task. Instead, I expect that CPSs are required to autonomously adapt their software and physical capabilities to changing task requirements. This research enables autonomous CPSs to adapt more naturally to external changes and leads to more dependable CPSs.

### 2.2 Facilitating informed decision-making about configurations

**Motivation**. I will enable developers to build configurable software by empowering them to decide confidently about configuration changes: for example, when they change an option, they ask whether their decision will affect any user, or create any **security issue or bugs** later down the line. Many options have various *constraints, consistency requirements, and dependencies* with other options. Such complexity makes it error-prone, or insecure: *e.g.*, a configuration change has been the root cause of the 30-hour Google cloud outage recently, or in another incident, a configuration change in one system component caused resource exhaustion in another component [9]. The main reasons behind these incidents are related to the deployment of erroneous configurations. This necessitates developers/operators to make informed decisions about configuration changes.

**My research plan**. I will give developers tools to get information about possible configuration dependencies, to automatically configure the software, and to update existing test cases. **My vision** is that tool support can reduce the technical debt and help developers to decide correctly whether to introduce or remove an option — although configuration errors may appear during deployment, the decision making process should be automated using appropriate tools. I will build systems support to facilitate decision making about introduction of options in the following aspects. First, I will conduct an empirical study to understand why an option appears and what information source developers use to decide about adding/removing options. I will perform the empirical study by interviewing developers that I identify via live monitoring of software repositories. Second, using the actionable insights from the empirical study, I will build tool support for measuring the impact of configuration options. For instance, I will explore the influence of introducing options on existing test cases. Third, to address the issue of misconfiguration, I will explore automatic generation of configuration patches, which satisfy the desired functionality while still maintaining the same level of performance. This research enables developers to make informed decisions about configurations. I am developing an **NSF** proposal to carry out this research and to support a PhD student that I will co-supervise with my current advisor, Christian Kästner.

**Expected impact**. This research intends to help developers and operators understand the potential impact of configuration changes to production. This requires addressing the fundamental challenges in understanding interactions between configurations, the deployment environment as well as potential workload. Therefore, I believe this research will enable companies to avoid big incidents such as the one happened to Google.

### Future Funding

In terms of future funding, my research at the intersection of machine learning and software engineering and distributed systems would be of interest to DARPA and NSF, as well DoD, or any other organization that is interested in cyber-physical systems (*e.g.*, DARPA call on Assured Autonomy).

## References

[1] **Pooyan Jamshidi**, Norbert Siegmund, Miguel Velez, Christian Kästner, Akshay Patel, and Yuvraj Agarwal, "Transfer learning for performance modeling of configurable systems: An exploratory analysis," in *Proc. Int'l Conf. Automated Software Engineering (**ASE**)*, ACM, 2017.

[2] **Pooyan Jamshidi**, Miguel Velez, Christian Kästner, Norbert Siegmund, and Prasad Kawthekar, "Transfer learning for improving model predictions in highly configurable software," in *Proc. Int'l Symp. Software Engineering for Adaptive and Self-Managing Systems (**SEAMS**)*, IEEE, 2017.

[3] **Pooyan Jamshidi** and Giuliano Casale, "An uncertainty-aware approach to optimal configuration of stream processing systems," in *Proc. Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (**MASCOTS**)*, pp. 39–48, IEEE, September 2016.

[4] Antonio Filieri, Martina Maggio, Konstantinos Angelopoulos, Nicolás Dippolito, Ilias Gerostathopoulos, Andreas Berndt Hempel, Henry Hoffmann, **Pooyan Jamshidi**, Evangelia Kalyvianaki, Cristian Klein, and others, "Control strategies for self-adaptive software systems," *ACM Trans. on Autonomous and Adaptive Systems (**TAAS**)*, vol. 11, no. 4, p. 24, 2017.

[5] **Pooyan Jamshidi**, Amir Sharifloo, Claus Pahl, Hamid Arabnejad, Andreas Metzger, and Giovani Estrada, "Fuzzy self-learning controllers for elasticity management in dynamic cloud architectures," in *Proc. Int'l Conf. on Quality of Software Architectures (**QoSA**)*, pp. 70–79, IEEE, April 2016.

[6] Soodeh Farokhi, **Pooyan Jamshidi**, Ewnetu Bayuh Lakew, Ivona Brandic, and Erik Elmroth, "A hybrid cloud controller for vertical memory elasticity: A control-theoretic approach," *Elsevier Future Generation Computer Systems (**FGCS**)*, vol. 65, pp. 57–72, 2016.

[7] **Pooyan Jamshidi**, Mohammad Ghafari, Aakash Ahmad, and Claus Pahl, "A framework for classifying and comparing architecture-centric software evolution research," in *Proc. of European Conference on Software Maintenance and Reengineering (CSMR)*, pp. 305–314, IEEE, 2013.

[8] Claus Pahl, **Pooyan Jamshidi**, and Olaf Zimmermann, "Architectural principles for cloud software," *ACM Trans. on Internet Technology (**TOIT**), In Press*, 2017.

[9] Matt Welsh. What I wish systems researchers would work on. http://matt-welsh.blogspot.com/2013/05/what-i-wish-systems-researchers-would.html.

[10] Claus Pahl, **Pooyan Jamshidi**, and Danny Weyns, "Cloud architecture continuity: Change models and change rules for sustainable cloud software architectures," *Wiley Journal of Software: Evolution and Process (**JSEP**)*, vol. 29, no. 2, 2017.

[11] Hamid Arabnejad, Claus Pahl, **Pooyan Jamshidi**, and Giovani Estrada, "A comparison of reinforcement learning techniques for fuzzy cloud auto-scaling," in *Proc. Int'l Symposium on Cluster, Cloud and Grid Computing (**CCGrid**)*, pp. 64–73, IEEE Press, 2017.

[12] **Pooyan Jamshidi**, Aakash Ahmad, and Claus Pahl, "Autonomic resource provisioning for cloud-based software," in *Proc. Int'l Symp. Software Engineering for Adaptive and Self-Managing Systems (**SEAMS**)*, pp. 95–104, ACM, 2014.

[13] **Pooyan Jamshidi**, ed., *BO4CO: an auto-tuning tool for big data.* 2016. https://github.com/dice-project/DICE-Configuration-BO4CO.

[14] **Pooyan Jamshidi**, ed., *TL4CO: Transfer Learning for Configuration Optimization.* 2016. https://github.com/dice-project/DICE-Configuration-TL4CO.

[15] Nithin Mathews, Anders Lyhne Christensen, Rehan OGrady, Francesco Mondada, and Marco Dorigo, "Mergeable nervous systems for robots," *Nature Communications*, vol. 8, p. 439, 2017.