Modern software systems have become highly-configurable, giving users the ability to customize how their applications behave, with access to hundreds or even thousands of preferences. This leads to the potential for millions or billions of program variants, each with a unique execution profile. While beneficial for the user experience, configurability creates problems for the software tester. The tester must validate that the system behaves as expected, since a lack of sufficient testing costs the worldwide economy billions of dollars annually. Research has shown that different configurations behave differently under the same tests cases hence testing a single instance of a configurable system is insufficient. Confounding the problem of testing configurability is that identifying and modeling the configuration space of many systems may not be that easy – with options often hidden under multiple layers of a system’s architecture, and implemented in different programming languages.

In this talk I first provide some insights into problems with configurability and show how it impacts our ability to efficiently and effectively test our software. I then discuss some state of the art techniques, which can help us to navigate this landscape efficiently. I also provide some empirical results that suggest there is locality where failures occur, and then demonstrate how we can leverage locality to develop self-adaptive software that reconfigures itself to avoid and guard against failures encountered in the field. I end the talk with a discussion of configurability outside of the software domain and discuss how we can use our techniques to benefit other scientific disciplines.