As software becomes more ubiquitous in systems of all types, so too does software engineering, even in fields where system developers have not traditionally considered themselves software engineers. Extending software engineering techniques – both by custom-creating novel concepts and applying well-established ones to new domains – to all people doing all types of system development is an area ripe for productive research. Though I have traditionally worked within safety-critical systems engineering (specifically in medical device interconnectivity), the techniques I've learned, adapted, and developed extend more broadly into other safety-critical domains (nuclear, automotive, etc.) and beyond into all fields where tool support and automation can improve the lives of those who interact with the development of software-based systems.

My doctoral work was initially focused on creating a software development environment for a specific type of medical application which governed clinical device interactions. While at first the project focused on standard software engineering technologies like model-driven development and automated code generation, the scope expanded to include architecturally-analyzable program aspects. As ensuring the safety of the developed system became increasingly important, and as I recognized the opportunity to include a systems-theory based hazard analysis technique in the development environment I was creating, I began to consider how development artifacts beyond code could be generated in a model-driven fashion. Seeing significant opportunities for formalization, I developed a pair of processes (one manual and one tool-assisted) which leveraged both the new, theoretically-grounded hazard analysis techniques and the deep architectural integration enabled by model-driven development.

Going forward, I am interested in continuing my work in three main ways. First, though I have established initial connections between my theoretical work and that of others both inside of and external to safety-critical software engineering, I would like to strengthen those connections by further developing the theories as well as performing detailed case studies. Second, I believe that the hierarchical nature of software-based systems can be exploited for significant developments in both the analytical power and ease-of-use of hazard analyses. Third, the compositional nature of modern systems represents a significant challenge to safety assessments, but a challenge that a formal grounding of certain key principles may be able to overcome. In all aspects of this work, I believe that grounding research in a formal or semiformal description of a system's architecture will not only achieve significant savings in developer and analyst effort, but will also act as a "sanity check" that will prevent analyses from becoming unhinged from the real-world systems they examine.