The pervasive nature of software (and the tendency for it to contain errors) has long been a concern of theoretical computer scientists. Many investigators have endeavored to produce theories, tools, and techniques for verifying the behavior of software systems. One of the most promising lines of research is that of formal specification, which is a subset of the larger field of formal methods. In formal specification, one composes a precise mathematical description of a software system and uses tools and techniques to ensure that the software that has been written conforms to this specification. Examples of such systems are Z notation, the Java Modeling Language, and many others. However, a fundamental problem that plagues this line of research is that the specifications themselves are often costly to produce and difficult to reuse. If the field of formal specification is to advance, we must develop sound techniques for reducing the cost of producing and reusing software specifications. The work presented in this dissertation lays out a path to producing sophisticated, automated tools for inferring large, complex code bases, tools for allowing engineers to share and reuse specifications, and techniques for developing new specification languages useful in modern engineering contexts. This dissertation introduces three main lines of research. First, I discuss a system that facilitates the authoring, sharing, and reuse of software specifications. Next, I discuss a technique which aims to reduce the cost of producing specifications by automatically inferring them. Finally, I discuss a specification language called Evidently which aims to make information flow security policies easier to write, maintain, and enforce by untangling them from the code to which they are applied.