How and when is software development an epistemological issue in digital scholarship? In both the sciences and the humanities, strategies for designing, building, and maintaining software increasingly interact with the justifications of knowledge claims. For example, in scientific computing, the overwhelming complexity of natural phenomena often requires simulation developers to reject the most theoretically principled designs in favor of unprincipled approaches that are, at least, computationally tractable. Similarly, in the digital humanities, the fundamental constraints of computing, such as the requirement to disambiguate knowledge representations, are often at odds with the basic tenets of humanistic inquiry – this tension sometimes leads to novel design strategies and technical interventions.

Software development has become a routine part of scholarly work, however our critical language for talking about software as a knowledge practice is lacking. Software development is conventionally understood as the practical matter of implementing specifiable computational tools, not as an ongoing process of materializing theoretical and epistemological orientations. An account of software that emphasizes epistemological issues is one that would direct attention toward the ways knowledge work inheres in code work; it would describe the means through which software becomes an object of knowledge. This project builds on theoretical work from science studies to conceptualize scholarly software as an epistemic object. The primary feature of epistemic objects, according to Knorr-Cetina, is that they are necessarily incomplete – this attribute is important because: "[o] nly incomplete objects pose further questions, and only in considering

CONSPICUOUS COMPUTING – SOFTWARE DEVELOPMENT AS A KNOWLEDGE PRACTICE

Seth Erickson | University of California, Los Angeles

objects as incomplete do scientists move forward with their work". The apparent incompletes or insufficiency of scholarly software can be thought of as a kind of conspicuousness. Two kinds of conspicuousness in scholarly software are posited: computational conspicuousness of scholarly objects and the epistemological conspicuousness of software objects.

Given this (necessarily sketchy and provisional) theoretical framework, the following research questions are put forward:

1. What are the kinds of conspicuousness that arise in scholarly software development? Can they be divided into the computational and the epistemological as presented above?
   a. What are the dynamics between these kinds of conspicuousness over time?
   b. How do the material aspects of scholarly and technical work structure the relationship between these types of conspicuousness?
2. How are practices other than software development used to manage the conspicuousness of scholarly software?

These questions will be addressed through a comparative case study of two groups of scholarly software developers: a group of physicists building computer simulations to study plasma phenomena and a group of humanities scholars building a web-based platform for authoring and publishing born digital scholarship. The case studies will be informed by both ethnographic fieldwork and historical research on the specific software practices observed at each site. The case studies will be carried out sequentially, one after the other, with roughly six months of fieldwork at each site.

Seth Erickson is PhD student at the University of California. He has a B.A. in History of Art and Architecture from Brown University and a MLIS from the Graduate School of Education and Information Studies at UCLA. His research interests are software studies, critical technical practice, digital humanities and science studies. Recent publication: Erickson, E.; Kelty, C.: "The Durability of Software". In: Kaldrack, Leeker (Eds.), *There is no Software, there are just Services* (2015).