Methodology
This poster employed the grounded theory approach (Corbin & Strauss, 1990), analyzing 40 interviews conducted with domain scientists (e.g., bioinformaticians, computational chemistry, theoretical physics) and computational technologists. Participants were from across the US (including CA, IL, IN, SC, MI, TX, etc.) and a small portion were from the UK (Scotland). Interviews were conducted either in person or by telephone. Following the interview guidance by an established protocol, the co-authors performed multiple iterations of data analysis and literature integration, yielding preliminary findings presented in this poster.

Findings
Throughout the coding process, three common themes were found within:

Tension of Specialized vs. Generalized
During the inception and design phase, developers may grapple with the tension of deciding to target specific or general tool. Although a domain-specific problem generally drives tool development, the generalizability of a tool will ultimately make a tool more appealing to adaptors across disciplinary domains.

• “[From a developer’s standpoint], “adapting these general tools to their specific domain to make it easier for the scientist to use because they often want some sort of custom organization or a button that does something very specific to the workflow, or something of that nature.”” (Researcher, TX, 12/16/14)
• “So there’s another way to look at it too, which is the software that underlies our project is generic, and other organizations have adopted that to create their own virtual organization, so that’s another way to look at the scheme.” (PI/Administrator, CA, 7/16/14)
• “The domain has to be specific, but keep their layers separate. So build it as a layered architecture so that the domain specific layer is there and then once you extract out all the domain out of it, all those tools generally— even though they are not fully baked, at least they are not tied up to a domain, but they are general.” (System Architect, IN, 11/18/13)

Tools Transferred Across Disciplines
As tools diffuse in the community, they tend to adapt for use in new and uses in virtual organizations, despite disciplinary differences. Often developers and adjust pre-existing tools to meet their needs, rather than building tools themselves. These changes can occur externally, as a tool jumps from one organization to another; however, the tool can also be expanded within an organization.

• “My-Proxy is a security tool for managing certificates and such, that was something that was developed at NCSA, but it’s been adopted by many other virtual organizations as this point in time... So as far as a tool being adoptable, it has to be specifically designed to be adapted for alternative uses.” (Administrator, IL, 7/16/14)
• “We pick the right tool for the job... the core focus for our work is always on application...And the other softwares have matured enough that we don’t have to do all that work over and over again. So we leverage what’s out there and try not to waste things from scratch.” (Senior Scientist, CA, 7/14/17)
• “Early on we brought in another postdoc whose expertise was in a different area, and she worked on adding that into the software package...so it was a way to extend it.” (Director/Chair, IL, 11/20/13)

Community Feedback
A tool is more widely accepted and adopted, the community of users around a tool will contribute changes, feedback, and general ideas that make the tool much more robust.

• “I think most successful tools have basically gone through a phase of development...when they worked with more users, they got an idea about what users want and they basically started catering to that, enabling others to use the tools.” (Senior Scientist, CA, 7/17/14)
• “Because what happens then is that people start using it, and then they start giving you questions, like: ‘I would like to do this.’ ‘Why doesn’t it do that’ or ‘this didn’t work.’ And you go back and you look at it, and you say, Oh, we never thought anybody would use it like that, and we don’t have an algorithm for that yet, but we have an idea... And we got just incredibly valuable feedback about what was important in the community, where was research needed, which we would not have gotten without people using the stuff we had done and giving us that kind of feedback.” (Administrator, IL, 11/20/13)

Conclusion
There are certain conditions in the community that affect the way a tool is reinvented or repurposed in different organizations, and ultimately diffused. Developers grapple with the commitment to make a tool either general or specific to domains. Kee and Browning (2010) argue that the approach to managing tensions in cyberinfrastructure development is not to choose one of the opposing poles, but to creatively embrace both simultaneously. In this community, developers argue that to choose one of the two poles is to resolve the tension that give rise to technologies, thus hampering the emergence of cyberinfrastructure. The management of tensions will also affect the likelihood of the tool transferring across disciplines. Open source computational tools are inherently user-driven and community-focused. The strategy for successful diffusion is to have a bias for action (Peters & Waterman, 1982) in other words, the community should champion a protyping feedback, thus promoting the diffusion process.

Given these conditions in the community, ‘permanently beta’ tools have emerged. Permanently beta refers to the simultaneous and collective development, which incorporates users into the process (Neff & Stark, 2003). This collaborative engineering values, are negotiated and incorporated in the design process and the products themselves (Neff & Stark, 2003). Multiple iterations lead to several versions of beta software throughout the ongoing process (Neff & Stark, 2003) being permanently beta allows for stakeholders to iterate, edit, and adjust as necessary.

In order for community members to address the tension between specialized or generalized tool development, they should abide by a standardized procedure regarding the development and tool architecture. This standardization will make a tool more appealing to adaptors across different disciplines, ultimately enhancing value in the scientific community. Complimentary to active development, community feedback encourages the evolution of tools; responsive tool tool developers constantly perform iterations and improve the tool over time, which makes the tools permanently beta and influence its likelihood of diffusion.

References

This material is based upon work supported by the National Science Foundation under Grant No. 1552860. The poster was presented on May 14, 2015 at the 2015 Chapman University Student Research Day, Orange, CA.