Scientific Research Software Ecosystems

John D. McGregor (PI), J. Yates Monteith, John E. Ingram
{johnmc, jymonte, jei}@clemson.edu
Strategic Software Engineering Research Group
School of Computing
Clemson University

The Sustainability Problem

The sustainment of scientific research software is both a technical and a business problem. There needs to be a market for scientific research software that defines the qualities desired in such software products. Here we present an analysis using Porter's Five Forces Model for Strategy Development, shown in the central figure. We will use MATLAB and related software as a case study for one route to sustainable research software. The five boxes in the figure correspond to the five forces that shape a competitive strategy. We address each below.

A Case Study: MATLAB

MATLAB grew out of the research and development of several numerical analysts in the late 1970s and early 1980s. Cleve Moler, Jack Little, and Steve Bangert founded MATLAB in 1984 to offer commercial versions of products that had evolved for several years. The product has evolved from a library of scientific routines, mostly for matrix manipulation, to support a wide range of basic mathematical functions which form the platform upon which a number of libraries are constructed. http://www.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html

Suppliers

The goal is sustainable innovation. One way to sustain innovation is to expand offerings by incorporating the work of other software groups. MATLAB has done this very slowly compared to consumer software. MATLAB was originally built on top of LINPACK and only many years later upgraded to the new LAPACK. A second way to sustain innovation is to build up from the base. MATLAB provides a platform for the design and development of libraries that are sold separately and that are small enough to evolve rapidly. Whether it is new features for the platform or for end customers, suppliers help drive an organization forward.

Buyers

Buyers want value and in the context of research this is very critical since research funds are usually hard to come by. Buyers shop for the features they need but don't want to pay for what they don't need. Huge, monolithic products often provide far more than a research project needs. Striking the correct balance of modularity and completeness requires analysis of domains and their needs. MATLAB provides the ability to define toolkits on top of the numerical analysis platform. Buyers can choose appropriate toolkits that meet their specific needs.

Substitutes

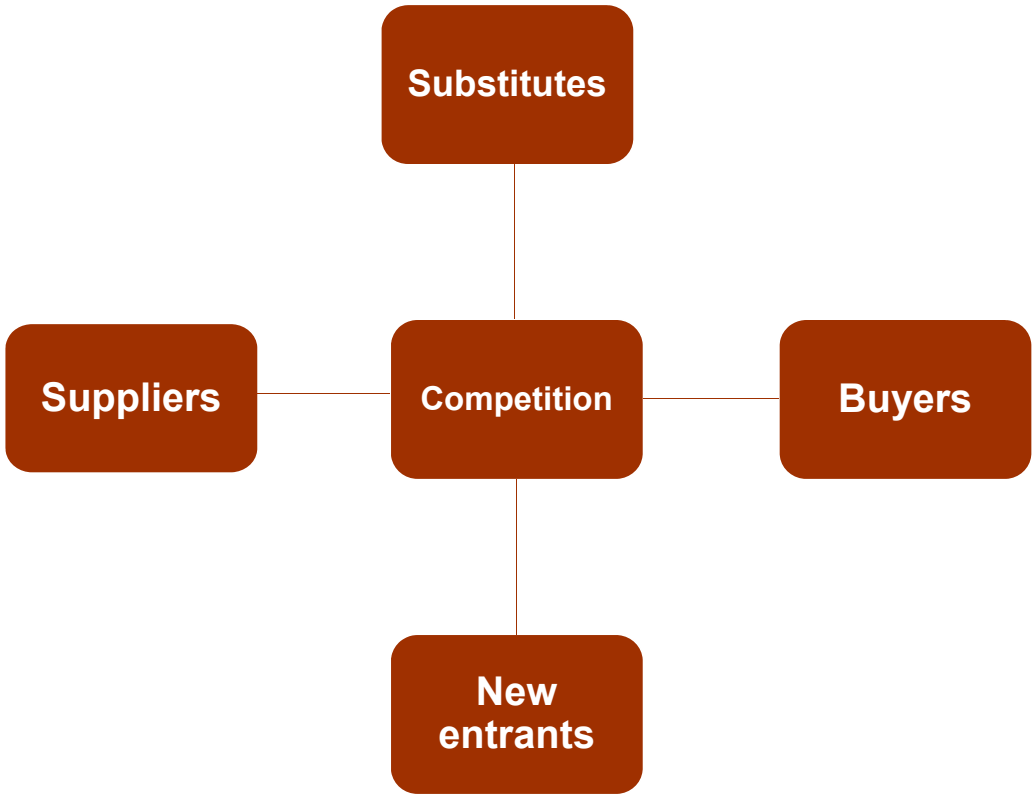
Those products that can be used in place of your organization's product drives feature selection and required qualities. Users of scientific research software can often find functionally equivalent alternatives in the vast stretches of the open source marketplace. What may differentiate products is the quality of the software. MATLAB users are interested in speed of computation and correctness of results. Choosing a lower cost alternative may require extensive testing to achieving the same assurance as using a well-known product such as MATLAB. Substitutes can mitigate this by providing test cases using well-know problems.

New Entrants

Much new scientific software is developed each year but most of it is never used outside a small circle of related projects. A new entrant into the market has substantial momentum to overcome if there are existing products that offer similar features. Particularly with open source software making large inroads into all facets of software. Research institutions are making ever greater efforts to commercialize research products but software is a very tough market. Most of the money made by a software company comes from auxiliary services rather than the product itself.

Competitors

Using a software ecosystem strategy in which you collaborate with competitors to build a common platform and then differentiate your product with extensions to the platform is cost-effective. MATLAB has many competitors, particularly open source competitors who either have a goal other than profit or a revenue stream to supplement their development.



Future Work

The commercialization analysis presented here is one aspect of the Business viewpoint in our previous poster about modeling software ecosystems. With research organizations increasingly interested in the commercialization potential of their products this type of analysis can provide useful input into the viability of maturing a product and provides guidance on how to structure the software to do that. .

Conclusion

Scientific research is a complex network of software, science and people. Producing sustainable, viable software presents an exceptional challenge to scientists and software engineers alike. By performing a strategic analysis such as Porter's Five Forces organizations can examine the commercialization potential for their scientific research software. We identified and discussed four factors involved in that success.

Model-driven, Incremental Development

Software development is moving away from hand coded programs to programs that are generated from models. The benefits include facilitation of iterative development since throwing away one version of the code for a newer one is simply a matter of regenerating the code from a revised model. This requires more attention to the abstractions that are defined. Modeling tools often include graphical editors that make design a faster and more interactive activity than writing lines in a programming language. These environments provide means of simulating the actions of the program very early in the development process. MATLAB has numerous building block libraries that provide a graphical view of the program elements and then writes the code that is executed.

Abstractions

Abstractions constrain the quality and scope of what can be communicated and analyzed in a domain. While exploratory research projects begin by exploring the unknown, periodically those projects should pause and consolidate results and define new abstractions upon which to build additional theory. Much like the platform nature of the product described below the abstractions should be extensible. Occasionally the tree of extensions is refactored as knowledge is built and new more powerful abstractions are identified. MATLAB's programming language makes a matrix a first class object.

Platform

The platform architecture style has facilitated the interaction among scientific researchers. Many research projects have been able to prototype new tools with less effort and expense by extending existing platforms such as Eclipse and MATLAB. It is also relatively easy to start an extension as a rough idea and evolve it to completion rather than doing a large amount of upfront design. Groups beginning to develop new tools should search for the most appropriate platform and use that to prototype and evolve a toolset. Simulink extends MATLAB but then is itself extended by numerous analysis plugins.

Architecture

Software architecture is central to a sustainable and extensible software product. Products such as MATLAB may start with a naïve architecture such as a library of functions but as the product tries to evolve a more sophisticated architecture is required to meet their business goals. MATLAB from its earliest versions used a façade to hide the FORTRAN underpinnings from users. Now there are front-ends for a number of different languages and adding new languages is a well-defined pattern. The addition of new features can be done by extending rather than changing the platform.