The Role of Customized Computational Tools in Product Development

Veena Tikare, Martin Heinstein and Steven Kempka
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185

Executive Summary

Computational tools, for the purposes of this work, may be broadly classified into two categories: (1) customized computational tools that analyze engineering performance of a particular product or process and general computational tools that have a broad range of capabilities that can be used to analyze the response of many different technologies, products and or (2) processes under many different situations. In this paper, we consider the former, customized computational tools that are highly developed for analyzing specific products for the purpose of improving the product or the process. Three examples of computational tools, which have been successfully integrated into product and process engineering, are presented. The critical features that make these three computational tools useful to product engineers are that they simulate the correct physical phenomena accurately and they are user-friendly. Product engineers, who are not expert computational analysts, can use them to evaluate numerous product designs quickly and easy. The ability to evaluate numerous designs results in improved products that take less time to engineer and often cost less. However, a consequence of computational tools with user-friendly, customized interfaces for advanced modeling and simulations is that the generality of codes to simulate many different products and processes is lost. Furthermore, the resources necessary to customize computational tools for analyzing a specific product or process can be very large.

This need to invest large amounts of resources to develop customized computational tools that are applicable only to a specific product or process has implications for managers of both computational tool development groups as well as product engineering groups. Computational tool developers must recognize that often product engineers do not analyze their product and process designs, which can benefit greatly by performance analysis, simply because the computational tools are too complicated and require too much experience and expertise. In such a situation, either the computational tool must be customized for that application or an expert analyst, who has the knowledge and expertise to correctly analyze the product or process, must perform the analysis. Conversely, product engineering managers must justify expenditure of resources necessary to customize computational tools for their particular application, or work with expert analysts, who can perform the analysis.

Abstract

Model-based computer simulations have revolutionized product development in the last 10 to 15 years. Technologies that have existed for many decades or even centuries have been improved with the aid of computer simulations. Everything from low-tech consumer goods such as detergents[i], lubricants[ii] and light bulb filaments to the most advanced high-tech products such as airplane wings[iii], wireless communication technologies[iv] and pharmaceuticals[v] is engineered with the aid of computer simulations today. In this paper, we present a framework for describing computational tools and their application within the context of product engineering. We examine a few cases of product development that integrate numerical computer simulations into the development stage. We will discuss how the simulations were integrated into the development process, what features made the simulations useful, the level of knowledge and experience that was necessary to run meaningful simulations and other details of the process. Based on this discussion, recommendations for the incorporation of simulations and computational tools into product development will be made.

Background

Without the aid of computer simulations, many of today's products would not exist. Laundry detergents are more concentrated but able to readily disperse in water without much agitation, able to clean more effectively while less damaging to fabrics, less polluting and requiring less water than detergents just a few years ago. Automobiles are safer, more comfortable, more fuel efficient, more reliable, have better traction, have fewer and less emissions, and have much longer service lives than one could have imagined just a few years ago. A single oi1 rig today can effectively drain areas that used to require tens and, in some cases, hundreds of oil rigs while improving performance in almost every way. All these technological advances in the last 10 to 20 years have largely been due to the use of computer simulations in their respective fields.

At the heart of product development is the ability of an engineer to understand the science that enables that technological improvement to the product. For example, the steam engine was invented long before Gibbs published his papers on thermodynamics. However, the understanding gained from the science of thermodynamics has given rise to continuous improvements and has led to the modern internal combustion engine. Science-based computer simulations facilitate this type of product development. The advantage of computer simulations is that they allow conceptualization and or visualization to a much greater degree than possible by closed-form analytic[#] techniques. The product engineer can explore many more engineering designs in more depth than possible analytically. Therefore, prototypes engineered with the aid of computer simulations have enormous advantages over prototypes that are engineered analytically leading to better final products.

While few dispute that integration of computational analysis into product engineering can

[#] We use analytic to mean closed-form solutions obtained without the aid of numerical simulations.

result in better products, designing and making available simulation packages to product engineers remains a contentious issue. Many computational analysts would argue that product engineers do not have the knowledge and experience to correctly use highly developed computational tools for analyzing their products. They feel that the proper use, application and results interpretation is sufficiently complex to lead to erroneous results and conclusions in the hands of a novice user. Similarly, many product engineers feel that simulation packages are unnecessarily difficult and complicated to use. They are discouraged from using the computational tools because the tools are not readily applicable to their products, engineers need to learn the vocabulary and syntax of the simulation package, the formats for inputting and outputting data are not standardized, the numerical solvers are confusing to use and other such difficulties.

In this work, we will discuss features of a computational tool that are useful to a product engineer. The paper will be organized in the following way: An introductory section will consider the process of developing computational tools. Next, a conceptual framework will be presented that will facilitate the understanding of the computational tool development. The third section will discuss specific computational tools that have been developed and are currently used in product development. These computational tools will be placed in the conceptual framework developed in the previous section. In the fourth and last section, we will discuss the features of the tools that made them successful and provide guidance on the development of customized computational tools.

Introduction

Before we consider what features of computational tools make them useful to engineers, let us briefly consider what expertise computational engineering scientists, known as analysts at Sandia and in this paper, and product engineers have and what roles they play in product development. Note that these are idealized summaries of their knowledge and capabilities and considerable overlap between analysts and product engineers may often exist in reality.

Analysts have extensive knowledge and experience using computational tools to perform numerical simulation in their specialize fields. They are knowledgeable about the models, numerical solvers, they understand the physics of the phenomena that they are simulating and they have experience with applying the simulations to study problems. They may be able to add to or edit the code to adapt it to a particular problem, and they can interpret the results of the simulations and recognize when the results are erroneous or faulty. However, analysts are typically not product engineers. While they are able to analyze a particular engineering design, they do not know the product, its manufacturing process, its function, the performance characteristics and properties that need to be optimized, engineering constraints, etc.

A product engineer is usually intimately familiar with the product, its function, its requirements, its engineering principles, how it is manufactured, the applicable manufacturing, regulatory and marketing constraints, etc. However, most product engineers are not experienced users of computational tools. Relatively small impediments

such as terminology, input and output file formats, documentation, etc. will derail efforts by product engineers to use computational tools. It is this critical problem of designing computational tools so they can be used by product engineers that will be addressed in this paper.

Conceptual Framework

For computational modeling to be useful to evaluate a particular engineering design, a product engineer needs a computational tool that will address the correct physical phenomena pertaining to the product. Next, the tool must allow him to interact with it easily. Among product engineers who use or would like to integrate computational tools into their engineering process, the most cited reason for not using them is that the computational tools are not user-friendly for engineering analysis and conceptualization[vi]. Product engineers improve upon engineering designs by making changes and evaluating how these changes influence product performance. After major engineering designs are evaluated, they are often refined by making small, incremental changes to look for performance trends. If the product engineer has to pass each small change to an analyst to perform the simulations and report back on the performance of that engineering design, not only can the process become cumbersome and slow, it can also inherently stifles the creativity of the engineer. In this situation where the analyst evaluates performance, the net result can be that the product engineer spends a lot of time optimizing the engineering by experimental and or analytic techniques[*], settles upon an engineering design and asks an analyst to evaluate its performance. Or the product engineers might forego the computational evaluation entirely and simply rely on experiments for improving performance to meet programmatic deadlines. The ability to refine the engineering designs and the ability to conceptualize the relationship between the designs and product can often be lost when the product engineer and analyst are out of sync. Customized computational tools allow product engineers to evaluate numerous engineering designs quickly and leads to improved prototypes for testing.

The conceptual framework presented in Figure 1 is used to describe computational tools. Here the horizontal axis represents the accuracy, fidelity and flexibility of the simulation, i.e. the "goodness" of the simulation package. The vertica1 axis represents the ease of use of the computational tool. In this representation as one moves to the right, the simu1ation becomes more capable of simulating a wide variety of problems with very accurate results. As one moves up, the computational tool becomes easier to use, allows the user to interact easily with the tool and requires the user to know little about the mechanics of the simulation. In such a framework, the region where analysts are necessary is the lower right region, where the computational tools are very capable and general, but requires experience and expertise to apply. The region where product engineers can function effectively is in the upper left area. Here the physics of the problem is simulated well, but the tool is easy to use and requires virtually no knowledge of the mechanics of the simulation. Only general engineering understanding of the problem and physics is necessary.

[*] Analytic is used to mean closed-form solutions done without the aid of numerical simulations

The observed tendency of computational tool evolution is that as the simulation contains more physics and fidelity, it tends to become less easy to use as shown by the red line in Figure 1. The most widely applicable computational tools tend to require experienced analysts who understand how the simulation works, which models it uses, how it applies the physics to solve the problem, etc. to apply the simulation properly to solve problems. Computational tools for engineers require that the tools be actively developed to make them easy to use while minimizing the loss of fidelity of the physics or the accuracy of the simulation results.

In the next section, computationa1 tools that are being used by engineers will be examined and described in the context the conceptual framework presented in Figure 1.

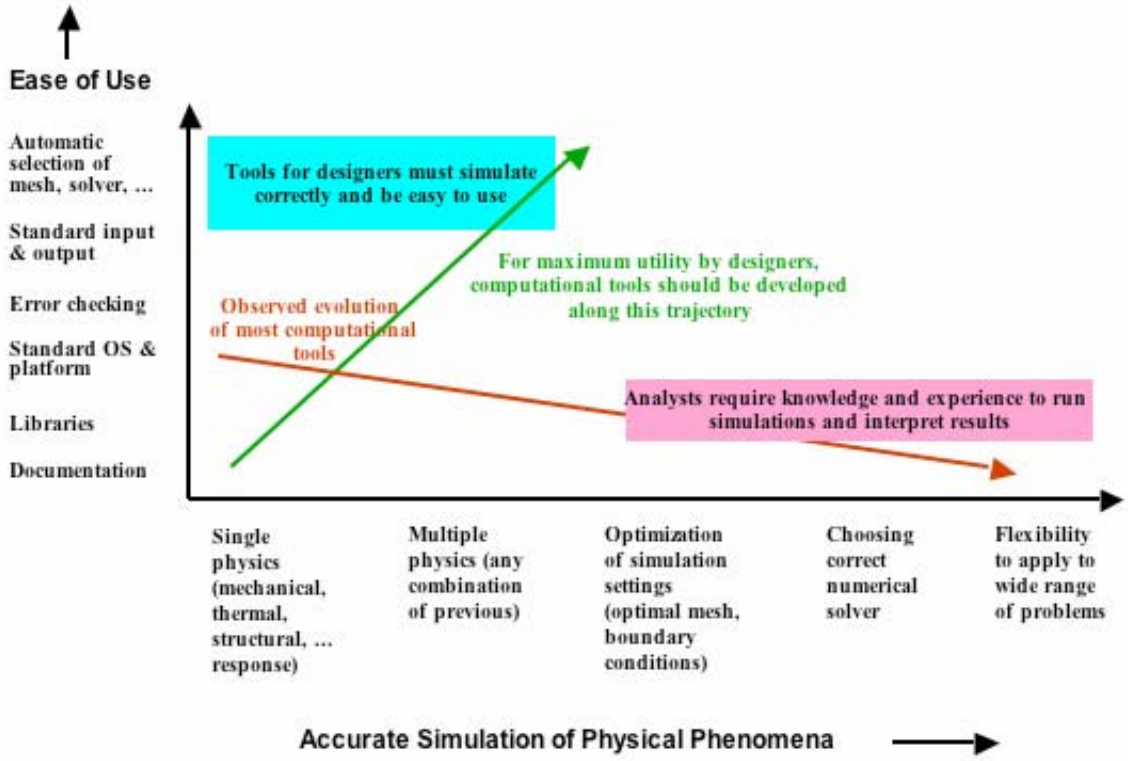

Fig 1. Conceptual framework for describing the utility of computational tools.

Computational tools used by engineers

1. LEX-D is a computational tool whose name is short for LIGA Exposure and Development. LIGA is an X-ray lithographic process for making metal or plastic micro-parts having overall dimensions up to several centimeters and feature sizes down to about a micron or somewhat less. In this process, a thick PMMA resist is first exposed to synchrotron radiation through a patterned absorber mask that defines the part geometry. The resist is then developed to remove exposed regions degraded by x-ray exposure, and this produces a mold that is filled by electrodeposition to produce metal structures. These structures may represent the finished metal parts or may serve as a tool for producing plastic parts by embossing or injection molding. Detailed descriptions of this process can be found elsewhere.[vii]

The models used to simulate exposure and development are complex, as they must simulate multiple physical phenomena occurring both simultaneously and sequentially. Simulating X-ray exposure of PMMA resists consists of computing the spectrum of X-rays produced by the synchrotron, the transmission and absorption these X-rays through beam filters, the influence of beam optics, and transmission and absorption in the PMMA. In addition to primary radiation, secondary radiation in the form of Auger electrons, photoelectrons and X-ray fluorescence must also be modeled. Some of this secondary radiation is absorbed in masked regions of the PMMA and so is important in determining dimensional tolerances of the developed mold. Simulating the development process consists of computing the evolution of a two- or three-dimensional dissolution front, taking into account the local absorbed X-ray dose, the development temperature and the transport of PMMA fragments. These models must address PMMA fragment transport in various feature geometries by mechanisms of diffusion, forced convection, natural convection, and acoustic streaming via sonic agitation.

Although the basic LIGA process is straightforward in concept, process improvement and optimization requires detailed understanding of the complex physics underlying each process step. LEX-D was developed to allow LIGA process (or product) engineers to quickly and easily evaluate the influences of all LIGA exposure and development parameters in order to improve tolerances, yield and throughput.

The models of the exposure and development processes are combined in the interactive computer program LEX-D to compute the required mask absorber thickness, exposure times, the resulting dose distribution, required period of development and expected part tolerances. It is also used to improve the process through analysis of mask membrane and PMMA substrate materials yielding reduced secondary radiation. This simulation tool has an interactive, menu-driven interface that does not require the simulator to know the underlying physics or models. Only an understanding of the process is required, as the code provides numerous help and warning messages to guide inexperienced users. LEX-D is usually supplied without a user manual. To make this possible, all of the process parameters are pre-populated (do not need to be input) with default values for standard conditions, and these are simply edited by the user as needed to evaluate the effects of any process parameters. In addition to making the interface easy to use, the

simulations take very little time, only a few seconds, on desktop computers with standard operating systems.

A rather unique feature of LEX-D is that input and output are displayed simultaneously, and "inputs" and "outputs" are largely interchangeable. That is, the user may specify as "input" either process conditions or desired results, so long as the combination produces a unique result. This is a critically important capability for most users of the code since, as engineers, they usually know what they want and are running calculations to find how to get it. To accomplish this, an intelligent shell on the core of the program monitors how the code is being used to discern the user's intentions. Where necessary, the shell performs root-finding, optimization or other numerical iterations to produce desired results that are specified as "input." The user can thus exercise the code in a manner of engineering analogous to using a closed-form analytical solution, rather than simply performing numerical experiments.

While LEX-D was originally intended for use by LIGA process engineers for process improvement and optimization, it has also proven useful to LIGA researchers and the technicians who perform the exposure and development. The combination of simulating the exposure and development process in detail, with complete relevant physics, quickly and easily has made this tool one that is used over and over by almost everyone directly involved in LIGA at Sandia.

2. SIP, Solder Interconnect Predictor is a computational tool that predicts the extent of fatigue degradation in electronic Sn-Pb solder interconnections. Solder joints are used to form package-to-circuit board interconnects in most electronic devices. They provide both the mechanical and electrical connection between the package and circuit board. During operation, the temperature of both the package and circuit board cycle from cool to warm and back. Both, the package and circuit board expand to different dimensions due to the temperature cycles placing cyclical stresses (known as thermo-mechanical fatigue) on the solder joints. Thermo-mechanical fatigue of solder joints leads to degradation and occasionally failure of interconnects.

Today, new package designs are being created while at the same time advanced package materials and circuit board laminates are being developed for applications that must endure harsher service environments and higher temperatures during the soldering process (e.g. new Pb-free soldering technology requires higher assembly temperatures). These rapid advanced in electronic package designs and materials are quickly causing existing empirically-derived reliability databases to become obsolete. The current quick pace of technology development and the shortened time-to-market will not allow the development of new reliability databases as it would be too expensive and time consuming to do the long-term empirical studies. Computational tools that can simulate the thermo-mechanical fatigue of solder joints and predict degradation are needed to allow the development of new electronic packaging technologies.

SIP simulates thermo-mechanical fatigue of solder joints for a variety of package types given the dimensions of the package, board and soldered interconnect, package and board

material, underfill and laminate, and temperature history. SIP predicts fatigue degradation using a microstructure evolution state-variable to modify the material properties in real time, thus capturing the continual changes to the microstructure as fatigue deformation accumulates in the microstructure. A finite element mesh of the solder joint geometry is automatically generated and used to map the fatigue deformation in the entire solder joint as a function of time. Developing SIP required extensive knowledge of meshing, mechanics, physical and mechanical metallurgy and solder technology. Unlike previous models, the models used in SIP were new models with improved fidelity and can address fatigue degradation everywhere in the entire solder joint.

While SIP has models that simulate many complex physical phenomena accurately, it was designed to be used by product engineers who are not well-versed in the disciplines of mechanics, metallurgy or soldering technology. To achieve this goal, the SIP code developers did the following: (1) They made the tool user-friendly by allowing the engineer to easily input parameters into the model using a graphical user interface which provides pictures of interconnections that can be evaluated with corresponding lists of acceptable parameter choices. They also made it easy to run simulations and generate thermal-mechanical fatigue lifetime predictions for Sn-Pb solder based on a couple of different failure criteria. (2) The time required to perform an analysis ranges from a few minutes to at most an hour on a desktop computer. (3) The software has the flexibility to address rapidly evolving technology of electronic packaging such as new geometries, new materials, different temperature cycles, etc. The software is structured so that it will be easy to add material models and failure criteria for new Pb-free solders as soon as the research has developed the material models and failure criteria for these new materials.

3. Eagle is a computational tool used to evaluate the engineering performance of pneumatic automotive tires. Over a century of pneumatic tire use for a variety of vehicles operating in varied conditions has resulted in many tires of different shapes, sizes, tread designs and internal constructions with each tire being composed of as many as 30 to 40 different materials in a variety of composite geometries. The, collective experience of tire engineers over the years has resulted in a wealth of empirical data on relationship between engineering parameters and performance. The challenge for tire engineers is to optimize tire performance by tuning these engineering parameters. A simple example is maximizing tire traction for a variety of road conditions, including wet, dry, oily, muddy and icy roads, etc., while minimizing wear, degradation due to fatigue, and cost. Gains in one performance measure will often result in a loss of another performance measure. State of the art tire engineering deals with a complex engineering space that is increasingly difficult to conceptualize. The ability to use computational tools within a rigorously defined optimization framework to explore tire performance has been very valuable to tire engineers.

Eagle is primarily a non-linear solid mechanics software package that can solve for the stress and strain state of a body under mechanical loading. Understanding how this general purpose finite element computational tool evolved from the lower right part of Figure 1 to a tool for tire engineers at the top left part of Figure 1 gives us insight into

'what makes a computational tool useful to engineers'. First, not all the models in the mechanics package were needed; just a few basic models were needed. While the underlying mechanics models for Eagle existed ten years ago, the numerical algorithms that allowed simulation of tire response needed to be developed. Once the basic modeling capability was incorporated in Eagle, a large effort called "productionization" of the code was needed to make the code useable by tire engineers who were not experts in computational modeling. One of the key efforts in "productionization" of the code was to construct simulation templates. These simulation templates restricted the computational tool to a specific, well defined simulation of a physical phenomenon. This approach allowed tire engineers to "speak their own language" in the specification of problem they wanted to simulate. They specified standard information about the tire such as tire dimension, tread geometry, materials and their arrangement – information they already specified in one way or another using established engineering practices. The practice of focusing on a particular, well defined simulation allowed the code developers and analysts to deliver a robust modeling approach, i.e., information from the simulation template could be used to automatically mesh the tire, choose the models and solvers correctly, set all simulation parameters in the computational tool correctly, run the simulation and present the results in formats that are meaningful to the tire engineers. In addition, the elapsed time to run most simulations is a few hours allowing engineers to have results overnight. This combination of accurate simulation of physical phenomena and accessibility to the complex software needed for the simulation via simulation templates has made Eagle an enabling tool for tire engineers. It has resulted in better tires that can be developed quicker and cheaper[viii].

Development trajectory in framework

From its conception, LEX-D was designed to be a tool that is used by LIGA process or product engineers. The evolution of LEX-D, over a period of almost seven years, occurred in a horizontal direction as shown in Figure 2. The interface was always easy to use, and the simulations always ran quickly. Over the years, a lead scientist, with support from several others, developed the increasingly complex physical models needed to accurately simulate exposure and development of the PMMA mold. As new models were developed, they were added to LEX-D to make it increasingly capable in areas such as secondary radiation, beam optics, and transport phenomena.

The models and the software package that Eagle used were highly developed more than 10 years ago, when Eagle development started. The software package could be applied to virtually any solid mechanics problem and was used to simulate the mechanical responses of a wide range of components made from many different materials under virtually any loading conditions. Eagle's software package was located in the far right, lower part of Figure 1 requiring an expert analyst was required to run the simulations. The decision to develop Eagle led to the unexpected trade-off by reducing generality in order to increase ease of use, as shown in Figure 2, by choosing models and developing solvers that apply specifically to tires and to specific, well defined simulations of physical phenomenon. The usability was greatly enhanced by allowing tire engineers to easily interface with Eagle via simulation templates and outputting results in formats that were easily

interpretable.

SIP uses the same non-linear solid mechanics solver that Eagle uses. The models that predict fatigue degradation were developed by many for scientific exploration. The metallurgical models to predict microstructural changes under cyclical loading were developed as a general capability. Solder technology and its use as interconnects has been an area of both computational and experimental research for decades. The original meshing capability was developed as a general capability that could be applied to any finite element analysis. While all these models existed and they could be applied to predict solder fatigue degradation, it required an analyst with specialized knowledge and experience in applying these models to generate a mesh, identify materials properties and input parameters, and analyze the output and interpret it for the electronics package engineer. SIP was created in less than two years by packaging all these capabilities to move vertically up as shown in Figure 2. The models and simulation capability existed, but packaging them and developing the customized, user interface that allowed application to solder joints with an automatically generated mesh, geometry selection, materials selection, etc. was a large effort.

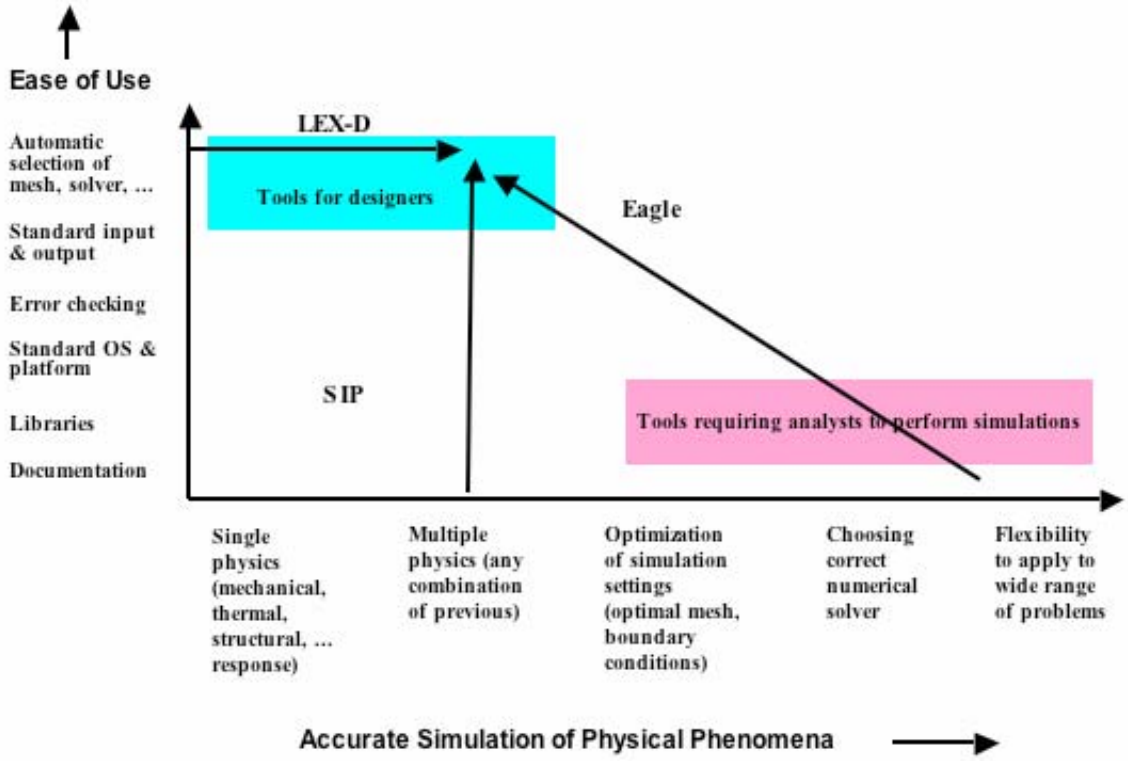

Fig 2. Trajectory of computational tool development for LEX-D, SIP and Eagle.

While all three computational tools presented in this section are high on the usability axis and to the left on simulation ability scale, the trajectories they took are very different. This illustrates that computational tools appropriate for product engineering applications may be obtained in a number of ways, even in the seemingly counter-intuitive trajectory of reducing simulation ability. In fact, this may be the most common trajectory for most customized computational tools as computational ability has evolved to a state where they can be applied very generally to a large variety of products.

Discussion

In all three examples, the models simulate the physical phenomena accurately using the latest scientific advances. LEX-D incorporates compact or analytical models to the maximum extent possible, and straightforward numerical methods for quadratures, root-finding or solving time-dependent ODEs are employed only where absolutely required. As a result, the code is able to simulate X-ray generation, secondary radiation and other physics very quickly. SIP integrated the latest understanding gained from non-continuum metallurgical modeling and simulation with non-linear solid mechanics to predict fatigue degradation in solder joints better than any other model. Eagle used the same highly developed non-linear solid mechanics software package, but had to develop numerical solvers for the accurate simulation of the physical phenomena. Tire engineers required that the output from Eagle be within 5% of the experimentally measured quantities.

The utility of all three computational tools is a direct result of the customized, user-friendly interface that allows the product engineer to evaluate engineering designs without being an expert analyst with extensive knowledge of modeling and simulation. Furthermore, they demonstrate successful implementation of properly designed software tools in the product development cycle. This component cannot be overemphasized. In all three cases, a lot of resources was devoted to making the computational tools readily useable by product engineers, who may be inexperienced computational analysts. If product engineers have to spend a lot of time learning how to run simulations and interpret their results, they will be reluctant to incorporate computational analysis into their product development cycle. If the computational tool is easy to use and can be applied readily to evaluate product designs, then product engineers will use it to evaluate new engineering designs and refine them.

A consequence of designing a customized computational tool that is readily usable by product engineer is that the generality of the models is lost. LEX-D has very advanced models for X-ray optics, generation and absorption, secondary radiation, dissolution kinetics and many other physical phenomena that can be applied to many other technologies; however the package LEX-D is specialized to LIGA and to the somewhat broader field of X-ray lithography. Similarly, the mechanical response models used in both SIP and Eagle are specialized for those applications. The general code used in SIP and Eagle can be applied to everything from the response of biological tissues such as bone to solders, tires, airplane wings and geological structures. However, as a tool that is useful to a product engineer, SIP and Eagle must be restricted to provide useful

information about a narrow set of technological problems. A simple example is that the same models in Eagle can also be used to study the performance of a rubber belt that drives a gear shaft. However, an expert analyst would be required to generate the mesh for the belt as the mesh parameters, such as mesh element shape and size would have to be adjusted for this particular geometry and loading. Furthermore, all the simulation conditions and parameters (i.e. mechanics model, material model, numerical solvers and output format) would have to be selected by an analyst.

Constant and extensive interaction between computational tool developers and product engineers is a critical component of ensuring that tools are developed to be useful to product engineers, i.e. the tools are "productionized" properly[♦]. The correct phenomena must be simulated under the operational conditions of that product. The input parameters into the code must capture the product parameters such as dimensions, materials, etc. correctly. The output from the simulations must be meaningful to an engineer. Models developed by scientists typically give a lot of basic information, which the scientist uses for exploration and discovery. However, a lot of information may overwhelm, confuse or be irrelevant to a product engineer. For example, the models in SIP can give three dimensional stress and strain tensors, grain size and temperature in the entire solder joint as a function of time. However, all this information is not useful to a product engineer, who is looking for a measure of fatigue degradation for that solder joint. Furthermore, the product engineer wants a measure that he can readily compare with other electronic packaging designs to assess the performance of that engineering. Understanding what product engineers want from a simulation is a critical component of making successful productionized computational engineering tools.

In all three cases presented in this work, the computational tools were designed by software developers working very closely with product engineers. In the case of LEX-D, the lead scientist wrote the software with users and their needs fully in mind. A scientist, who understood the LIGA exposure and development process so well that he could anticipate product engineers' needs, wrote most of the code in anticipation of future needs. A small team of scientists and software specialists developed SIP. One member of the team was an experimental materials scientist who had worked extensively developing and testing solder joints for electronic packages. Thus he was able to guide the development of the tool and the interface, so that electronic package engineers could interact easily and get meaningful results readily. A large team of computational mechanics experts, software specialists and tire engineers developed Eagle. The mechanics experts, software specialists and tire engineers were in daily contact during the tool development via E-mail and telephone. Due to the large size and technical diversity of the Eagle team, they also scheduled a week-long meeting every six months to ensure that the customized computational tool, Eagle, facilitated tire development.

Having stated that the simulations results should give results that a product engineer wants, it is also important to optionally provide more information that tire engineers may find useful. For example, one of the outputs from Eagle is the frictional energy

♦ Productionized properly is used to mean that the emphasis is on the software product not on the computational science.

dissipation (the slip distance of the tire relative to the road multiplied by the normal stress and friction coefficient). The frictional energy dissipation is related to the wear that the tire experiences and is measured in standard tests that tire engineers use to evaluate tires. However, simulations can provide much more information that an experimental test cannot. The simulation can provide the normal stress everywhere in the footprint of the tire. The simulations can also provide the resulting stresses at important locations within the tire that indicate potential problems, e.g., where stress is concentrated at levels that may damage the material.

The resources, both in time and money, necessary to package models into customized computational tools applicable to specific products can be very large. This level of resource expenditure can be justified only if gains made by using the computational tool are commensurately large. LEX-D was developed as part of the entire LIGA technology development effort. The resources spent to develop LEX-D were a small fraction of the resources spent to develop LIGA technology. For a relatively small expenditure,10 to 15% of the LIGA development expenditure[ix], LEX-D enabled the processing of PMMA molds to progress much faster than the other processing steps required to make LIGA parts, resulting in excellent dimensional control of the mold and thus of the final LIGA component. SIP was developed by a team of scientists and software specialists to meet the demands of electronic package engineers and is used extensively to evaluate designs for solder joints in electronic packages, which are ubiquitous. Furthermore, advances in soldering technology are occurring at a rate that is much faster than ever before with new geometries and new materials. SIP has allowed these new materials and new geometries to be evaluated at a much faster rate than possible either experimentally or analytically. Eagle was developed as a joint effort between Sandia and Goodyear by large team of scientists, software specialists and engineers over a period of 10 years. Eagle was used approximately 18,000 times last year by Goodyear to evaluate various tire designs. This has resulted in better tires while reducing the time and money required for these new designs.

The need to invest large amounts of resources to develop customized computational tools has implications for managers of both computational tool development groups as well as product engineering groups. Computational tools developers must recognize that often the computational codes that they have developed are not used in product and process development because product engineers, who are not expert computational analysts, do not have the knowledge and experience to apply computational tools. This may require the development of customized computational tools, assigning an expert analyst to do the analysis or a combination of partial customization and training the product engineers. Product engineering groups need to recognize the cost and effort involved in developing customized computational tools. They must either justify the expenditure of resources to develop customized computational tools, work with an expert analyst or invest in training a product engineer to acquire the skills necessary to apply general computation tools for their engineering process.

Summary and Conclusions

In this work, we have considered the role of customized computation tools in product and process development. Customized computational tools are tools that are developed to be used by product engineers, who are not expert computational analysts, to analyze specific product designs. Three customized computational tools were examined in detail. In each case, the features of the tools that allowed them to be successfully integrated into product design were ease of use, quick analysis time, correct and accurate physics, and the meaningful results for product designers. This level of customization allowed product engineers to analyze large number of engineering designs easily and quickly resulting in improved products that require less time to engineer and often cost less.

However, this level of customization was achieved at a price. The generality of models to simulate many different problems is lost; customized computational tools are readily applicable to only limited physical phenomena under narrowly defined conditions. Development of the customized tools required sophisticated and prolonged interaction between product engineers, software engineers and computational scientists with large amounts of resources devoted to the development. The need for investing large amounts of resources to develop customized computational tools that can improve products, but are narrowly applicable presents management challenges to both product engineers as well as computational tools developers. The decision to invest to develop customized computational tools must be made jointly between computational tool developers and product engineers to assure that the necessary resources are devoted. Furthermore, product engineers must justify the expenditure of these resources.

Acknowledgement
We wish to thank Timothy Trucano and Harold Morgan for critiquing this paper and the many suggestions they provided, Stephen Lott, Karen Jefferson and Michael Hobbs for their helpful suggestions, and Stewart Griffiths, Michael Nielsen and Dale Moseley for information on LEX-D, SIP and Eagle, respectively.

---

[i] E. Grald, "Meeting the Wide-Ranging CFD Needs of Materials Processing", www.Fluent.com/about/news/newletters/02v11i1/s1.htm; D. Nicolaides, "Modeling and Optimization of Detergents," Proc. of the 5th World Conference on Detergents, Pages 177-181, AOCS Press, (2002).

[ii] www.prod.exxonmobil.com/scitech/technology/mn_synthetic.html;, Editors, **Surface Modification and Mechanisms** edited by G.E. Totten and H.L. Dekker, New York (2004) 756 pp; K. Konno, D. Kamei, T. Yokosuka, S. Takami, M. Kubo and A. Miyamoto, "The development of computational chemistry approach to predict the viscosity of lubricants," Tribology Int., 36[4-6] 455-458 (2003).

[iii] J.B. Vos, A. Rizze, D. Darracq and E. H. Hirschel, "Navier-Stokes solvers in European aircraft design", Progress in Areospace Science, 38 [8] 601-697 (2002); A list of links for aircraft models is provided at http://jsbsim.sourceforge.net/fslinks.html.

[iv] The Fourth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, July 2001, Rome, Italy; Proceedings of the Cermic Interconnect Technology, April 2004, Denver Colorado; T. Rappaport, K. Kosbar, W. Tranter,

and K. Shanmugan, **Principles of Communication Systems Simulation with Wireless Applications**, Prentice Hall, (2004).

[v] **Rational Drug Design**, edited by A.L. Parrill and R. Reddy, ACS Symposium Series 719 (1999).

[vi] Personal communication with A. Muyshondt in July 2004; S. Holswade in August 2004; B. Cole August in 2004; T. Aselage in September 2004; R. Fellerhoff in September 2004; C. Vanecek in September 2004; P. Chavez in September 2004.

[vii] E. W. Becker, W. Ehrfeld, P. Hagmann, A. Maner, D. Munchmeyer, "Fabrication of Microstructures with High Aspect Ratios and Great Structural Heights by Synchrotron Radiation Lithography, Galvanoforming and Plastic Moulding (LIGA Process)," Micro-electronic Eng., 4, 35-56, 1986; D. Munchmeyer and W. Ehrfeld, "Accuracy Limits and Potential Applications of the LIGA Technique in Integrated Optics," Pro-ceedings of the SPIE, Micromachining of Elements with Optical and other Submicrometer Dimensional and Surface Specification, 803, 72-79, 1987; W. Ehrfeld, P. Bley, F. Gotz, J. Mohr, D. Munchmeyer, W. Schelb, H. J. Baving, D. Beets, "Progress in Deep-Etch Synchrotron Radiation Lithography," J. Vac. Sci. Technol. B, 6, 178-182, 1988.

[viii] Personal communication with Dale Moseley, Manager of Tire Physics and Computational Mechanics at Goodyear in March 2005.

[ix] Personal communication with S. Griffiths in January 2005.