Are Your Models Agile, or Fragile?

Rick Donnelly, *Parsons Brinckerhoff, Inc.* (donnellyr@pbworld.com)

November 4, 2009

Abstract

The increasing sophistication and capabilities of innovative travel models bring with them more complexity, reflected in both the models themselves and their software implementation. Like their predecessors, these models are typically developed in a linear manner after their initial specification is completed. While arguably appropriate for simpler models with large experience base, there is ample evidence that this approach is failing to deliver newer models within the scope, schedule, and budget prescribed for them. A new approach to modeling — agile development — is based upon its successful use in the software industry. Two recent success stories using this approach for developing innovative transportation models is described to illustrate its wider potential.

In my almost 25 years of experience in building travel models we have typically employed a linear model development process. It begins with a model specification, which often becomes part of the scope of work for a consulting contract or agency work program and budget. Developers then begin work on the model, an activity that can include data analysis, model estimation, its implementation in software, testing, calibration, validation, and application. Near the end of the project documentation and a final working model emerge, which are sometimes critically examined by peers. In an ideal world these events coincide with the schedule and cost originally envisioned. In the real world I have seen this fortuitous alignment occur occasionally, usually when building relatively small models employing familiar methods.

Unfortunately, more often than not most contemporary travel model development projects experience some degree of failure. Most do not fail entirely, with abandonment of the pursuit. However, it has become increasingly more common for projects to be delivered late, over budget, or with scaled back capabilities. This appears to be especially true for new and innovative developments, such as activity-based travel demand, integrated land use-transport, and region-wide DTA models. In fact, it is difficult to think of a single such project in recent years that has been delivered within scope, schedule, and budget. In a recent review of innovative models no instances were found of such successes (Donnelly et al. 2009). Developers appear to routinely err in their estimates of the time, data, knowledge, and resources required to deliver innovative models. Moreover, the resulting complexity has often overwhelmed customers attempting to "get their heads around" the delivered data and models. Such failures raise legitimate doubts about the viability and affordability of advanced models, which in turn will limit their adoption and application.

There is an interesting parallel to similar experiences in the software development industry. The field is much larger, and the products developed range from ad hoc software written by end users

to highly complex and feature-laden mainstream operating systems and enterprise applications. Thus, it is difficult to make definitive statements about their success rates, as there are literally tens (if not hundreds) of thousands applications in the marketplace. However, the literature abounds with statistics about the high failure rate — measured in terms of project delay, cost overruns, or reduced functionality — in the information technology and software development realms. Ewusi-Mensah (2003) reports that most major software development projects experience some degree of failure. Similar outcomes are reported by Atwood (2006) and Verner et al. (2007).[1]

While the causes of these failures are myriad (as they are in travel modeling), a common theme in most of them involves the overall development approach itself. The "big design up front" (BDUF) that typifies linear development has been faulted in many ways. Some argue that once the model specification becomes embedded in the contract that design cannot change to meet changing expectations or new knowledge (Lindvall et al. 2002). Others complain that the emphasis on a single final deliverable often translates into long development cycles. Delays that occur during them are often insidious, becoming apparent only towards the end of the cycle (Martin 2008). These shortcomings can arguably be overcome by more vigilant project control measures, although such becomes far more difficult when using geographically dispersed development teams. However, they cannot overcome the more difficult problem of uncertainty surrounding user requirements. It is quite common for clients to not fully understand user interface needs or how the model will be used until they actually apply it. The BDUF approach typically cannot respond to such experience, as the project is typically complete when it is ready for use by the end user.

Agile development (AD) techniques were devised to overcome these and other problems. There are several variants of AD in practice (Martin 2002). Most stress short incremental development cycles, typically of a month in duration. The work typically begins with "the simplest thing that could possibly work," a concept often attributed as Occam's Razor. Working interim models are produced at the end of each cycle, giving the client that ability to start testing and evaluating the model far earlier than typically possible. The development team meets with the clients at the end of each cycle to review progress, become familiar with the interim release, and agree upon priorities for the next cycle. Though lacking all of the desired functionality, the interim releases allow the client to guide the development of the software capabilities and user interface. Moreover, the improvement in functionality resulting from gained during subsequent cycles can be readily tested and critically assessed.

Our initial goal was to use Scrum, one of several AD approaches, to improve our practices in software development and to produce code with higher reliability. It quickly became obvious, even before we started achieving success in that realm, that the same mindset and principles could be usefully applied to the design and development of innovative travel models. This realization came at a pivotal moment when there was discontent and disappointment, at vary levels, over how some leading-edge modeling projects had turned out. Most had one or more components that had taken longer or cost more than budgeted to complete, and in a few instances, significantly more so. Some wrote this off as the inescapable cost of being a pioneer, and a consequence hardly unique to travel demand modeling. Others, including the author, were unwilling to accept that innovation and effective project management and timely delivery were fundamentally at odds.

[1] Statistics specific to scientific and engineering applications were not found in the literature, although several notable failures have been widely reported, to include several aerospace projects (Heckt & Buettner 2005) and the aborted update of the national air traffic control system (Charette 2005).

Our first application of Scrum was undertaken in conjunction with the Oregon Travel and Land Use Model Integration Program (TLUMIP). A second generation model design was completed in 2001, and a multi-year BDUF contract was executed to deliver it. Several years of model development followed, during which time other tasks and studies not related to development were also undertaken. The key components of the model form the column headers in Figure 1. Because of its modular integrated structure all of the components had to be completed in order to obtain any model functionality. Looking again at Figure 1, it meant that all of the intermediate steps shown were skipped, and the ultimate versions shown as shaded boxes at the bottom had to be made operational. Most were completed on schedule, but some of the novel and more complex models took longer than expected, delaying the entire project. Opportunities to use the model for key projects, which would have ensured continued support from upper management, were lost as a consequence. Once finally assembled the resulting model system was fragile, as many of the interfaces between the components had been developed in isolation by different developers.

In hindsight it was felt that an agile approach would have resulted in an interim operational capability much sooner. Though lacking many of the desired capabilities the experience gained with the model would have been highly instructive. The testing and evolution of several components could have been substantially completed in advance of the others. Experience gained using the interim model could have informed ongoing development, and priorities might have changed. Users would have provided feedback on the user interface earlier, and would have gained insight into desired functionality that comes only through repeated use of the model.

The transition to an AD approach for the TLUMIP work came midway through the project, when the limitations of the traditional approach became apparent. Fortunately, it was able to provide benefits despite its late adoption. It became clear that a number of intermediate steps towards the ultimate specification would have been possible, as illustrated by the boxes in the middle of Figure 1. Starting with quick response models with synthetic parameters would have provided a rudimentary modeling system capable of incrementally moving towards the ultimate design. Indeed, some of the components that were incomplete at the time of the transition were converted to this incremental cycle. Others not originally envisioned, such as the environmental models (far right column in the Figure), have since been added. Weekly team meetings, a key part of Scrum, are held using web conferencing, and the project directors meet quarterly to assess progress, resolve problems, and decide upon priorities for the next three months. While objective measures appear elusive, there is a higher level of satisfaction on the part of both the client and development team with the visibility, accountability, and flexibility of this new approach.

The experience gained in Oregon heavily influenced the approach taken to develop a statewide travel demand model for Maryland. This project was designed with an agile mindset and progress from the outset. The eventual goal is to develop an integrated model of the economy, land use, transportation, and the environment similar to what has been accomplished in Oregon. However, it was anticipated at the outset that such a model would need to evolve in several phases, with some phases lasting one or two years. While structured quite differently[2], the same functionality shown in the column headings of Figure 1 are desired. The components are being incrementally developed

[2]The Maryland model is a multi-level model, where long distance travel is microsimulated at a regional level and internal trips are modeled at a more detailed statewide level. Most of the work to date has focused on the transport side of the model, whereas a great deal more emphasis has been placed on the economic and land use components in Oregon.

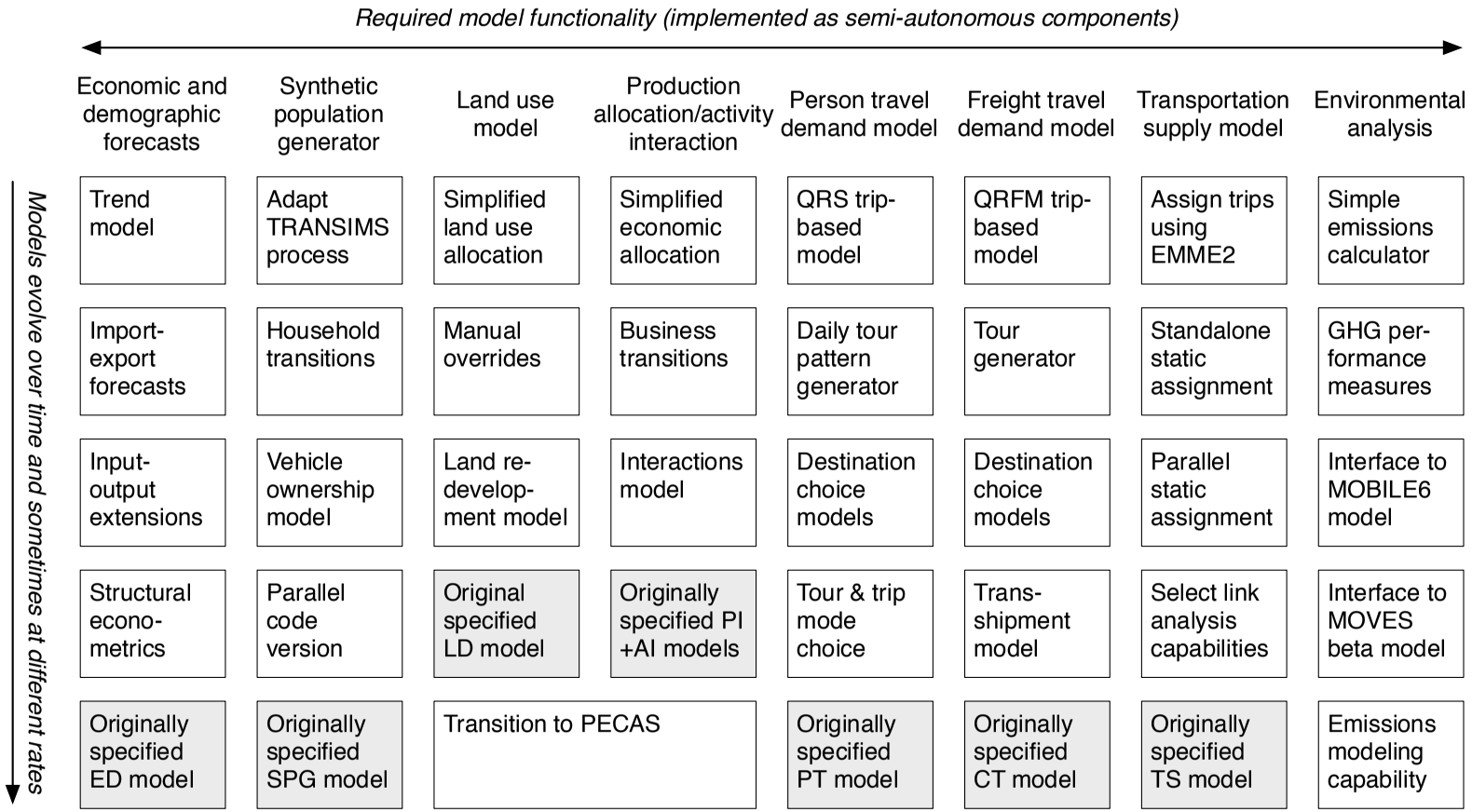

Figure 1: TLUMIP model development structured as an agile development process

in the stages shown. As a result of taking this approach the first generation of the Maryland model was completed within one year — on time and budget — and is undergoing validation and early applications at the time of this writing. The model does not yet meet the calibration targets set by the developers, but is surprisingly robust, and certainly within the range of accuracy attained by many other statewide models.

The next steps in Maryland will depend somewhat on the experience gained through application of the interim model. It is expected that in some cases that the first generation formulation of some components might perform well enough such that further work on them will be deferred, while the need for devoting more resources to other components will become apparent. It is important to note that the even progression of all components shown in Figure 1 is notional. There might be large gaps between some components, while others have more intermediate milestones. The important point to be made is that the work program and priorities are flexible enough to be guided by experience and knowledge gained along the way.

It is readily acknowledged that AD is not a panacea, and that other factors played a large role in the success of the Oregon and Maryland models. Several criticisms of AD appear to be equally applicable to attempts to craft agile model development approaches. Some criticize the agile mindset as being little more than "muddling along" without any assurance of reaching the ultimately desired product. This outcome can be avoided by careful elaboration of the use case[3] at the outset, which catalogues the desired functionality. It requires discipline on the part of the team leader(s) to keep these long-term goals in focus, rather than allowing the team to lose momentum by making a series of small and inconsequential tweaks to the initial product. The agile approach obviously will not work if it is opposed by the developer or client. Moreover, some clients (who may even see value in it) complain that their funding for a model is a one-time investment. Their management will either not support an incremental approach, or will decide that an early interim version is "good enough" without an understanding of the implications of using an incomplete model. It is not clear how to overcome such obstacles.

It might not be necessary for developing systems using mature technology, as adequate knowledge exists about the costs and risks associated with their deployment. However, those types of models are not what this conference is all about, nor what many in attendance are trying to achieve. The potential of the innovative models under discussion at this and related conferences have been widely documented and enjoy growing acceptance among researchers and practitioners. Adopting an agile modeling mindset will enable the pioneers to reduce the risk and uncertainty associated with such pursuits, and permits the assimilation of rapidly expanding knowledge about such models into existing development projects. Given the paucity of funding for research into and deployment of such models it is imperative that we be as efficient as possible in the limited number of opportunities we have. This prescription is not radical, as an incremental path from trip-based to activity-based models has been encouraged by others (Vovsha & Bradley 2006, Lemp

[3] A use case is used in software engineering to capture the functional requirements of the system. It focuses upon information, constraints, and requirements flowing through or interacting with the system. It is often depicted through examples of use or illustrated with several scenarios. In the context of transport modeling the use case would describe how the user would apply the model, to include specification of inputs, analytical capabilities (the types of questions to be posed to the model, and at what scale), the outputs required, interaction with the model (graphical and non-graphical user interfaces), and interfaces to external databases, models, or processes. In short, it describes how the model will be used rather than focusing on its mathematical or theoretical structure. The latter, by contrast, is the primary focus of most model specifications.

et al. 2007). AD takes this idea one step further by supplying a formal and proven framework to implement such models, thereby enhancing their likelihood of success. While arguably not best suited for every innovative modeling pursuit, the early success of the agile modeling approach in Maryland and Oregon commends it to developers elsewhere.

References

Atwood, J. (2006), 'The long, dismal history of software project failure', `http://www.codinghorror.com/blog/archives/000588.html`. Viewed August 19, 2009.

Charette, R. N. (2005), 'Why software fails', *IEEE Spectrum* **42**, 42–49.

Donnelly, R., Erhardt, G., Moeckel, R. & Davidson, W. A. (2009), Advanced practices in travel modeling. NCHRP Synthesis 40-06 interim report.

Ewusi-Mensah, K. (2003), *Software development failures*, MIT Press.

Heckt, M. & Buettner, D. (2005), 'Software testing in space programs', *Crosslink* **6**, 31–35.

Lemp, J. D., McWethy, L. B. & Kockelman, K. M. (2007), 'From aggregate methods to microsimulation: assessing benefits of microsimulating activity-based models of travel demand', *Transportation Research Record* **1994**, 80–88.

Lindvall, M., Basili, V. & Boehm, B. (2002), 'Empirical findings in agile methods', *Lecture Notes in Computer Science* **2418**, 81–92.

Martin, R. C. (2002), *Agile software development: principles, patterns, and practices*, Prentice Hall.

Martin, R. C. (2008), *Clean code: a handbook of agile software craftsmanship*, Prentice Hall.

Verner, J., Evanco, W. & Cerpa, N. (2007), 'State of the practice: an exploratory analysis of schedule estimation and software project success prediction', *Information and Science Technology* **49**, 181–193.

Vovsha, P. & Bradley, M. (2006), 'Advanced activity-based models in context of planning decisions', *Transportation Research Record* **1981**, 34–41.