Preface

Purpose

This text presents a broad reformulation of how software-based products are realized. At its core, this begins with a reframing of how singular (one-size-fits-all or one-of-a-kind) products are developed, based on a reconsideration of the elements of conventional approaches. It then extends this modified approach as the basis for a systematic approach for building sets of similar products. This portrays a disciplined industrial approach that envisions how complex high-quality software-based products can be engineered and manufactured at significantly reduced cost while being customized to fit diverse and changing needs. The motivating vision is the ability to mass produce and evolve customized software-based products.

Objective

This work traces back to long-ago observations that similar software capabilities were repeatedly being independently reproduced. At the level of software components, this is less true today, but products as a whole are still largely developed as "unique" solutions, whether for a single customer or for a broader market, to perceived opportunities for automation. Furthermore, current approaches result in products that become increasingly difficult and costly to modify as needs change. The concept underlying a better approach is that similar problems are amenable to similar solutions; this provides the basis for building customer- or market-targeted products that are both customized to fit specific needs and more easily modified as those needs change over time.

Context

When software first came into use with the emergence of computers in the 1950's, its capabilities were a limited and narrowly focused contribution to targeted operational endeavors. Software was seen as a simple means of automating tedious clerical and

computational tasks and enabling more responsive monitoring and control of simple mechanical devices and processes.

Today, software is an essential and pervasive enabler of endeavors in every facet of business, government, and daily life, from controlling industrial, energy, aerospace, and military systems and devices to enabling aspects of agriculture, finance, healthcare, education, entertainment, communications, and transportation. Even such categories are an insufficient characterization of the role of software in that it also controls the underlying interconnecting framework that increasingly ties all of these together into a functioning holistic ecosystem.

Despite this essential role, continued use of conventional development practices results in software that is overly complex, costly, flawed, and difficult to change. The underlying weakness in this approach is that it was conceived on the premise that each product is unique and unprecedented, requiring trial-and-error development from essentially first principles. It becomes a constraint and limiter on the ability of individuals and enterprises to be effective and efficient and adapt to changing needs.

Software practices have improved enormously over the last 60 years, through numerous attempts to bring greater discipline and predictability to software development and automation of routine tasks. These efforts have resulted in improvements in productivity and product quality but none have produced a fundamental advance given the prevailing narrow mindset of how software development should work.

Content

For a variety of reasons, software providers are wary of changes in development practices. Change brings uncertainty and a perception of increased risk of inferior results verses following conventional practices. However, the inherent flaws in a conventional approach also increase cost and limit a product's value over its potential useful life. A time will come when the inherent costs and risks of the conventional approach will yield to a more disciplined, more flexible manufacturing-based approach. An alternative approach, based on the related concepts of program families and mass

customization properly applied, addresses many of the issues that plague the conventional approach.

The conventional view of a software product is as a text that is repeatedly modified to express a changing understanding of a problem and its corresponding solution. An alternative view is based on the idea that every realizable product is an instance of a universal natural set of products. This set is further partitioned into a hierarchy of subsets based on a concept of behavioral similarity. Each such subset is characterized sufficiently to determine whether a needed product is a proper member of that set. When the need for a product is first considered, it is not known exactly which instance of the universal set is to be built. As a product is developed and revised over its useful life, every interim realization of that product can be seen as being a different instance. Similarly, when a product is later revised to meet changed needs, it becomes yet another instance of that set as well.

This alternative approach arises from an engineering conception of a product as a holistic realization of envisioned behavior rather than a reductionist focus on the devising of elements that in some combination happen to produce that behavior. This text is not prescriptive as to the detailed practices of software development but instead provides a descriptive formulation of an approach based on a theory and concepts of software development as a coherent discipline.

The objective here is to coherently describe the overall concepts for the engineering and manufacture of software-based products built upon a familiar base of known (but selectively revised) basic practices. This text relies on but does not discuss specific practices related to programming, design or testing techniques, tools, nor the specifics of software methods (except as needed to provide guidance on certain unconventional aspects of the proposed methodology).

This text is organized as follows:

The Foundations and Future of Software Production

Defines a vision toward reconceiving the development of software-based products as an engineering and manufacturing discipline, leading to significant improvements in productivity and product quality

Basic Software Product Engineering

Takes a different look at building a singular but changing software product

Domain-specific Engineering

Explores the concept of a product family as the basis for a systematic approach to the engineering of an envisioned set of similar products, enabling the manufacture of customized instances as needed

Automating Software Production

Describes a progression of formulations for automating mechanical aspects of the engineering and manufacture of software-based products

Anticipating Change

Surveys emerging concepts and techniques that could contribute to attainment of the proposed vision, its further evolution, or even a future reconception

Audience

This text is a systemic guide to a unified industrial approach to producing software-based products and systems. It provides both organizational, management, and technical perspectives on the adoption and practice of this approach within an enterprise. It is not a guide to the various detailed practices of software development, which are fully explored in a vast array of other publications and continue to evolve. Rather, it assumes a moderate degree of familiarity with the practices involved in a conventional approach to building software, preferably based on having actually developed and modified software and having experienced the difficulties of doing so. An understanding of the challenges of organizing and managing a software project or enterprise would also be beneficial. Lightly adapted forms of conventional practices are a basic foundation for this approach but these are described only sufficiently to explain

how they fit within the approach, particularly focusing on any differences from conventional uses. Without some reasonable familiarity with these practices, it can be difficult for the reader to appreciate the described alternative or to envision how a complete practice of this approach is achieved.

Acknowledgements

The overall concept presented here and many of its detailed elements have been developed and discussed in numerous papers and presentations over the last 40 years. A variety of enterprises, in government and industry, have beneficially applied tailored versions of this approach that have enabled them to be more responsive to their business and customer needs. This text is an attempt to tie all aspects of the approach together into a coherent whole, elaborating and refining several beyond past descriptions.

This text is the culmination of my long experience during the emergence of the theory and practice of software engineering. In particular, this has included experience in the 70's on a variety of commercial development efforts, projects in the 80's and early 90's at Software Architecture and Engineering (Software A&E) and the Software Productivity Consortium (SPC), working in the late 90's as an independent consultant to government and industry in the U.S. and Europe, and working in the 00's on the staff of the Carnegie Mellon Software Engineering Institute (CMU/SEI) in support of improving software engineering and acquisition practices of U.S. government agencies. In every case, I have benefited from the knowledge and expertise of numerous co-workers whose contributions to the substance of this text are too extensive to list in detail but were critical to the ultimate result.

Of greatest value to this work were

- Dr. David Parnas and other contributors to the Naval Research Laboratory/Software Cost Reduction project (NRL/SCR) [1981-82]
- Andrew Ferrentino and my co-workers on the Spectrum project at Software A&E (notably, Richard McCabe, Dr. Tom Shields, and Vicki Florian) [1981-88]

- Judd Neale, Dr. Art Pyster, and my co-workers and customers of the Synthesis project at the Software Producibility Consortium (SPC) (notably, Richard McCabe, Dr. David Weiss, and Dr. Stuart Faulk) [1989-96]
- Participants in the Thomson-CSF (Thales) corporate reuse initiative and supporting European Software Institute (notably, Pascal Maheut) [1996-98]
- Staffs of the CMU/Software Engineering Institute (SEI) Software Product Line Acquisition initiative [1998-2002] and Acquisition Support Program and associated customer programs (notably, Brian Gallagher, Terry Dailey, and Alfred Schenker) [2002-12, 2015-19]
- Matthew Campbell (of Hypar) and James Kirby (of SPC and the US Naval Research Laboratory), whose many helpful perspectives and insights have greatly improved ideas presented in this text

In addition to these primary influences, there have been numerous other communications on a wide variety of topics that have influenced my views. In cases of specific significance, particular publications are cited herein. I fail to cite many others that have been less of an influence on me personally or are outside the scope of this work (or that I have simply missed: the relevant literature is vast and growing). The contributions of all are sincerely appreciated, beyond any overt acknowledgement.