to bridge the abstractions in the domain analysis and architectural design phases with the concrete realizations of these abstractions in the implementation and maintenance phases. In the analysis and design phases, patterns help to guide developers in selecting from software architectures that have proven to be successful. In the implementation and maintenance phases they help document the strategic properties of software systems in a manner accessible by developers at many levels of experience.

If software is to become an engineering discipline, the successful practices and design expertise must be documented systematically and disseminated widely. Patterns are important tools for documenting these practices and expertise, which traditionally existed primarily in the minds of expert software architects. Without a thorough understanding of the patterns underlying domain-specific architectures, design, and implementations, OO software reuse will remain a largely unfulfilled promise.

Schmidt is a faculty member at Washington University. He has successfully applied patterns-based software reuse techniques in large-scale projects involving telecommunications switch management, network management for mobile communications systems, and electronic medical imaging systems on high-speed ATM networks. Schmidt has published widely in IEEE, ACM, USENIX, and IFIP technical journals and conferences. He is editor-in-chief of the C++ Report magazine, where he co-authors a column on distributed object computing. In addition, Schmidt is a co-editor on a recent book on design patterns entitled *Pattern Languages of Program Design* with Jim Coplien of AT&T Bell Labs.

Organization Domain Modeling: A Tailorable, Extensible Framework for Domain Engineering
Mark A. Simos
Organon Motives Inc., Belmont, MA.
mas@organon.com

Organization Domain Modeling (ODM) is a highly tailorable and configurable domain engineering method, useful for diverse organizations and domains, and amenable to integration with a variety of software engineering processes, methods and implementation technologies. The method offers a systematic, exemplar-based approach to analysis of commonality and variability within both legacy systems and requirements for new systems. Under funding by the ARPA STARS Program, ODM has been extensively documented in a guidebook (Version 1.0) [1], as well as in shorter papers [2]. The guidebook provides a formal process model (documented in IDEF-0), workproduct descriptions and templates, and detailed domain engineering guidelines.

ODM is structured in terms of a core domain modeling process, which can be tailored via a set of supporting methods and optional extensions. The core ODM lifecycle formalizes an intuitive process that can be applied to a wide variety of domains (including non-software related domains) and on a very small scale (e.g., the entire modeling phase could be prototyped within a few meetings). The method as currently documented, however, is intended to address software-intensive domains within large-scale systems.

ODM reflects the idea that domains are socially defined agreements about an "intended scope of applicability." Domains are always grounded in some "organization context," which might be a company or division, a consortium of multiple organizations or a community of interest such as a standards organization. Stakeholder issues, always a potential problem in any project, turn out to be critical risk factors in domain engineering, which by definition involves designing for multiple contexts of use. The ODM process uses systematic exploration of this stakeholder context to guide selection, scoping and definition of domains strategically aligned with the business interests for the organization(s).

Evolution and Applications. ODM grew out of the design of the Reuse Library Framework (RLF), a STARS-funded technology effort that applied structured inheritance network-based and rule-based knowledge representation techniques to support domain modeling. The method was later refined in close collaboration with Patricia Collins of Hewlett-Packard Company's Software Reuse Initiative, and formed the basis of an HP-proprietary workbook and method used successfully on a number of internal projects [3]. The most extensive application of ODM to date has been on the Army STARS Demonstration Project, in combination with RLF and the Conceptual Framework for Reuse Processes (CFRP) [4,5]. Since completion of the guidebook, several projects (e.g., Rolls-Royce, Logicon) have conducted trial applications almost entirely on the basis of the guidebook alone; this has served as a kind of "stress-testing" for the documentation (reports not yet available). A final guidebook revision (under STARS auspices) is slated for completion in 2Q1996.

Types of Systems Addressed. Some methods assume that domains span the entire scope of a related family of applications, and that a single generic architecture can
be discovered (or imposed) on that family of applications. While not excluding such monolithic domains, ODM facilitates definition of sub-system level domains. In ODM, the area of functionality covered by the domain of focus may encompass the entire scope for a family of similar systems (a "vertical domain" with respect to those systems) or cut across only a sub-portion of the systems (a "horizontal" domain with respect to the systems). Furthermore, the "intersection" of the domain focus for any system could correspond to a delineated sub-system or to distributed functionality (e.g., security requirements). Domain modeling can therefore be approached in an incremental way that facilitates iterative re-engineering of portions of legacy systems as well as guiding opportune architectural choices for newly engineered systems.

Processes and Workproducts. Conventional wisdom tends to divide domain analysis into phases addressing the problem space and solution space respectively. Since the problem space/solution space distinction is key to most systematic software development life cycle models as well, this approach inevitably embeds binding commitments to preferred or implicit software life cycle models and software design methods. ODM is structured into three main phases: Plan Domain Engineering, Model Domain, and Engineer Asset Base. The transition from domain model to asset base is not a problem to solution space transformation, however, but rather a selection of a subset of the variability defined by the domain model, for a defined multiple set of customers. This is a fundamental distinction between ODM and most other domain analysis methods.

ODM phases are iteratively decomposed into sub-phases and tasks; each task in turn is documented via sequences of or alternative activities and a structured set of workproducts that integrate information gathered throughout the entire process. Major results of each phase: domain definition, domain model, and asset base; are targeted to provide direct benefit to the project stakeholders as well as providing a systematic starting point for the next phase.

The Plan Domain Engineering phase includes a number of "upstream" tasks in domain engineering, many of which are not formally supported in other methods: identifying project stakeholders and objectives, and strategic domain selection from a candidate set of "domains of interest." The domain definition process iterates between a set of exemplars and a set of defining rules to help uncover implicit contextual assumptions in the domain scope, and to identify related domains.

The Model Domain phase produces a Domain Dossier, Domain Lexicon, and a set of concept and feature models, the format and content of which are more dependent on the particular supporting methods chosen. Our greatest experience is with RLF-based models, and there is a rich body of experience emerging on specific modeling techniques and principles for model organization that best support the ODM approach.

The integrated domain model (including the concept and feature models) captures both the commonality and variability of a given set of applications. Modelers distinguish the set of contexts of practice relevant for software systems in their domain, including, at a minimum, the developers' and end-users' environments (the "system-in-development" and "system-in-use" contexts, in ODM terms). Variability across multiple system-in-use contexts is usually echoed by a process or design decision in the developers' context, so ODM provides criteria for selecting from a wide variety of data elicitation techniques, including artifact analysis, interviewing of domain informants, and process observation. A current ARPA-funded task involves integrating scenario-based elicitation techniques from the SEP methodology as part of an ODM supporting method for data elicitation, and applying the composite method in the health-care applications area [6].

Each domain has a own unique pattern of such contexts, which might include maintenance, field installation, or user customization. Each context is a "site" where particular system features of interest may be bound. For example, early spreadsheet applications combined spreadsheet creation and data entry functions in one run-time program. Vendors created a powerful innovation by introducing "spreadsheet compilers," which split these two capabilities (each an ensemble of many related functions and features) across distinct contexts. (This pattern recurs in many other kinds of applications, such as DBMSs and high-end document management systems.) ODM's Extend Domain Model task includes a repertoire of such patterns and techniques for systematic exploration of such "gaps" in the domain "feature space."

In the Engineer Asset Base phase, a subset of the variability encompassed by the domain model is selected, based on identification of specific customer applications. The asset base architecture sub-phase addresses, not generic system architectures in the conventional sense, but the architectural tradeoffs directly relevant to supporting variant feature combinations, via parameters, families of components, generators, decision support applications for processes, etc. The process is thus applicable to engineering of
product or process-oriented assets from any portion of the software life cycle (requirements, test plans, procedural knowledge). System architecture methods can be invoked here as supporting methods. For example, a current ARPA task is implementing an ODM supporting method for building GenVoca-style asset base architectures, utilizing a suite of tools developed as part of the STARS Army Demo project [7].

Tool Support. Organon Motives, Inc., in partnership with WPL Laboratories, Inc., and with partial support from an ARPA STARS grant and contribution from Loral Defense Systems-East, is currently re-engineering RLF to be more comprehensive and accessible. Originally implemented in Ada83, has been ported to Ada95 (utilizing the widely available GNAT Ada95 compiler), and re-architected to utilize World Wide Web- and Object Request Broker (ORB)-based technology to support a Web-accessible browser/editor interface. The new OpenRLF system will provide an extensible framework for building domain models and applications that navigate these models, providing key support technology for ODM-based domain modeling.

References


Mark Simos is president of Organon Motives, Inc., a software consulting and engineering research firm specializing in domain engineering methods (ODM), tools (RLF), and reuse strategic planning. Mr. Simos first learned the "craft" of building reusable components and application-specific languages as an applications specialist in the computer typesetting field. He has been an active participant in the reuse field since moving into research in the mid-80's, and has authored or co-authored many technical and working group reports and published papers on reuse and domain analysis. He was initial chief programmer on the STARS RLF project, is the principal developer of the Organization Domain Modeling (ODM) method, and co-authored several other STARS reuse documents including [CFRP93]. He holds an MSE from University of Pennsylvania. He is also a well-known songwriter and fiddler in traditional folk music circles.

Domain Analysis, Domain Modeling, and Domain-Specific Software Architectures: Lessons Learned

Will Tracz
Loral Federal Systems Owego, NY
tracz@lfs.loral.com

In 1992, the ARPA-funded Domain-Specific Software Architecture (DSSA) program Avionics Domain Application Generation Environment (ADAGE) project developed a domain analysis process for creating domain-specific software architectures.

This process was based on the STARS domain analysis process, developed by Ruben Prieto Díaz, and FODA (Feature-Oriented Domain Analysis), inspired by Shalom Cohen at the Software Engineering Institute. The