

Amnon H. Eden. "Foreword". Ch. in: Toufik Taibi (ed.) *Design Pattern Formalization Techniques*. Hershey, USA: Idea Group Inc (forthcoming).

Software design is a fledgling discipline. When the 'software crisis' came to be acknowledged during the late 1960s, software development projects have been marred by budget overflows and catastrophic failures. This situation has largely remained unchanged. Programmers still create poorly understood systems of monstrous complexity which suffer from a range of problems directly linked to the lack of abstraction: Lack of means of communicating design decisions, absence of effective pedagogic tools for training novice programmers, and inadequate means for maintaining gigantic software systems. Recent years witnessed an explosion of loosely-related software technologies, techniques, notations, paradigms, idioms, methodologies, and most of all proprietary and poorly-understood ad-hoc solutions, driven by market forces more than by design, planning, or research.

Design patterns were introduced to programming practices at the end of the 1980s as a result of dissatisfaction with software's state of affairs. The few means of abstraction in existence at the time, such as algorithms and data structures, narrowly suited procedural programming, poorly fitting with the growing use of object-oriented programming. For the first time, an abstraction technique at hand was general enough to be useful for practitioners and academics alike, specific enough to enter textbooks, broad enough to be useful during any stage in the development process, and generic enough to support any programming paradigm. The introduction of design patterns marks a turning point in the history of software design.

In 1995 we witnessed the publication of a catalogue (Gamma, Helm, Johnson & Vlissides, 1995) of twenty-three design patterns written by four experienced object-oriented designers. The catalogue, which came to be known as the 'Gang of Four' catalogue, was an immediate success. The abstractions described offer a rich vocabulary for conceptualising, designing, brain-storming, communicating, documenting, understanding, maintaining, and teaching about software. Each pattern captures a design motif that is common enough to deserve wide recognition, described in clarity and sufficient detail to indicate the consequences of choosing to apply it. Patterns help novices to avoid common pitfalls and encourage experienced programmers to build better software. As a result, design patterns entered textbooks and became the subject matter of scientific papers, conferences, and intensive effort for providing tool-support by a broad range of industrial software development environments. In the decade since they have entered the zeitgeist, design patterns have revolutionized software design.

Early on, the attempts to reason about and provide tool support for design patterns have led many to recognize that verbal descriptions and case studies are not enough. The software engineering community came to realize that conceptual clarity and automation require a formal specification language. The central

problem in software design has therefore shifted from seeking suitable abstractions to providing precise means for capturing and representing them. But existing modelling notations, which were designed for documenting design decisions tailored for specific programs, proved inadequate for the purpose of modelling abstract design motifs. This shortcoming has motivated the investigation in formal modelling techniques.

Mathematics is the most successful conceptual tool for capturing, representing, understanding, and using abstractions. Effective mathematical analysis and modelling is the hallmark of modern science and a mark of maturity of an engineering discipline. The research in formal techniques for modelling design patterns is therefore the next natural step for software design. This line of investigation is vital for achieving conceptual clarity and ever more potent means of abstraction. This book provides a summary of this investigation.

A convergence of formalization techniques for design patterns, expected to evolve within a decade or two, is vital for establishing a sound foundation for using and understanding patterns. Convergence is also crucial for communicating about and teaching patterns. Given the central role of design patterns, such an achievement is widely taken to be the most important vehicle of progress for software design and a prerequisite for a regimented engineering discipline. The next generation of software design techniques may very well depend on accomplishing convergence. We hope that this book shall speed and facilitate this process.

Amnon Eden
Layer-de-la-Haye, September 2006

References

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Systems*. USA: Addison-Wesley Professional.