Foreword

Engineering organizations developing large complex systems are usually not capable of determining an "overall optimal" system design. Rather, the system is divided in "components" or subsystems (such as an axle in a car or a module in a software product), for each of which a performance can be measured, an optimal design can be found or at least approximated, and for which a designer (or engineer or team of engineers) is responsible. Each engineer then makes, at first, decisions to optimize "his" component. In real organizations, designers often develop considerable pride in the solutions they have found for their components.

However, it is the very nature of complex systems that the components cannot be optimized in isolation, but that they interact in determining the quality of the overall system (via space constraints, or via the exchange of fluids, air, force, electricity, or information). To some degree, these interactions are known from experience and can be anticipated, or are embedded in accepted design principles. However, in any complex design project that is not entirely routine and marginal, many such interactions are not known at the outset. Engineers find out about them at design reviews, prototype construction milestones, system integration tests, or sometimes in informal conversations. The interactions then force them to (sometimes grudgingly) concede compromises of "their" component design in the interest of the whole. But the adaptation is (almost) always local, in the direction of system performance improvement, not global as the global performance function is not known.

Some empirical studies suggest that systems become much harder to develop as they get bigger. It is a common experience of project managers that their project "iterates" through multiple cycles without convergence - the team starts with a certain approach, then makes modifications, which are then discarded, and at the end, they come back something they had tried before. This has, for example, become widely known in the software industry, where the current trend is to strictly limit the size of development projects. However, the reason for this problem has not been clearly explained. Moreover, the known remedies are very restricted: (a) modularize the system into independent subsystems that can be developed in parallel without interactions, and (b) communicate frequently among team members so no one works on a design based on obsolete information.

This thesis presents an analytical model of a New Product Development (NPD) organization as a complex problem solving system, which makes two contributions: first, the model explains clearly why larger system size makes development exponentially more difficult. Second, the model outlines possible remedies that are applicable in practice.

The complex system is modelled by a network of nodes. Each designer makes periodical design decisions (after some problem solving), taking into account his latest knowledge about the neighbouring components. The design decisions are communicated to the other designers periodically, that is, with a delay (this is typical practice - changes are not communicated immediately because people do not have time, because they are not aware of the ramifications, and sometimes because they do not like to communicate anything "halfbaked" before they know it is right). Whenever one designer changes his component, he changes the context for his neighbouring designers, who may then also have to change their designs to have the best solution (in some organizations, this is called "snowballing"). The system has reached an equilibrium (an accepted solution) when no designer wants to change his design any longer in attempting to improve his component.

As the system grows, the interactions lead to cycles (as one designer changes his decision, others have to also change because their boundary conditions have changed, which in turn forces the first designer to change), forcing the system into oscillations and ultimately into divergence toward extreme and bad solutions (unless the team stops and re-starts). These fundamental dynamics of complex systems explain the above-cited empirical observations, including the basic remedies of modularity and frequent communication.

The thesis then analyzes four managerially relevant variations of the base model, using simulations as the analysis method as there are no closed-form solutions for these more complicated cases. The first is cooperation among engineers: suppose all engineers care not only about their own component, but each one has the capability of calculating the total effect of his design decisions on the whole system (on all other components - this is the opposite extreme of the base case). The simulations show that problem solving performance improves (systems oscillate less, converge faster and diverge less often), but not radically; the fundamental problem of increasing system size persists. This is because the designers make their decisions still based on partially obsolete information, as they hear about other designers' decisions not immediately, but after some delay.

The second variation is immediate broadcasting of all design decisions: imagine each engineer posting his latest design status every evening on a central blackboard, and all engineers reading these status reports every day. This is again an extreme case unfulfilled in practice (even in the days of 3D-CAD systems, as communication costs become prohibitive when designers try to stay abreast of all developments in the system), but instructive. It turns out that cooperation among designers combined with immediate broadcasting effectively controls the negative effects of large system size and almost completely suppresses the problem solving deterioration.

The third variation is the use of preliminary information: surprisingly, going slower by the individual engineer can help the system to reach a solution faster. That is, if a designer does not implement his "optimal" decision right away, but goes only part of the way toward that solution, he sends a signal in the "right direction" to the others, without dislodging their decisions as much. The current "optimum" is likely to become obsolete anyway as other designers change their component designs. As a result, the overall problem solving dynamics for the system improve (up to a limit - if everyone goes infinitely slow, the system will also slow down).

The last model variation is ignoring links: in desperation, the team may overlook, or ignore deliberately (perhaps in hope that the links are not so important), some of the interactions among the components. A trade-off is the result; the speed of convergence to a solution increases (as de facto system size is reduced), but the quality of the solution becomes worse.

The model in this thesis shows precisely how a rugged landscape arises from seemingly innocuous components, through their system interactions (the shift of the optimal decisions and of the achievable performances at the component level, caused by decisions at neighbouring components). Here, the rugged landscape is not a metaphor, but it is caused by mechanisms that realistically (the assumptions are even slightly optimistic) model the dynamics in real projects.

The results make an important theoretical contribution to an improved understanding of the fundamental levers that project managers have in large projects. In addition to the traditional levers of frequent communication and modularity, there are other measures such as strict coordination (at least across subsystems), immediate broadcasting of important decisions, preliminary information, and cutting interactions if the project is extremely urgent.

We find that this thesis links three only partially connected literatures, those of NPD, organizational design, and complexity theory. In doing this, the thesis significantly goes beyond existing methods and adds to the knowledge in the management of complex projects in a significant way.

Prof. Dr. Arnd Huchzermeier	Prof. Dr. Christoph Loch
Vallendar, August 2002	Fontainebleau, August 2002