Outline of this book The Leverage Principle Healthy systems reproduce Disease, decay, and erosion Understanding technical debt
Multiple types of debt Best practices The difference between reuse and leverage How to Build Software that Lasts Design People

Best Practices Technical Debt Deferred problems Below the water-line Practical reality Debt is risk Types of debt What Is Bit Rot ? The crisis The cure Automate your testing Refactor ruthlessly Fix all known problems Make changes incrementally Core practices Automatic system test Unit testing Development loop Strategic version control Best Practices for Leverage 1. Make reuse a fundamental element of your software planning 2. Use a four-layer architecture 3. Create an architectural model based on reusable components 4. Build your entire design around core interfaces 5. Create a Domain Specific Language for your most important interfaces 6. Create standard protocols for interacting with components 7. Build a reusable application for your specific domain 8. Encapsulate business rules with a standard way to apply them 9. Automate all core interactions with your application 10. Use Diff Testing to accelerate your test creation and maintenance

Build a culture of Leverage Culture and world view Belief systems Revolution Call to action What is culture? Culture happens Crafting your culture Return on investment How to build the culture you want The impact of culture Essential elements of culture

Learning - Managing knowledge to optimize results Learning is a Strategic Capability New era with new challenges 50 Tricks Philosophy Focus on learning not training Each person is unique and learns in different ways Learning and promotion Skill Mapping Dreyfus model for learning new skills Manage a map of skills in your organization Reward new skills Build a learning culture Best practices for group learning Build experts Learning scorecard Invest in tomorrow Cost and Benefit Analysis Balance learning and doing Switching technologies Learning must be a priority Technology Map Levels of Skill Master Skills Competent Skills Beginner Skills

Definition Planning your leverage Hope is not a plan What not to leverage Build a Lifecycle of Leverage Your Lifecycle May Limit your Leverage Economics of Software Reuse Learning from Reality Lifecycle Phases Complexity management Strategies that work Measurement - Metric to leverage by You get what you measure Technical debt limits leverage Technical debt is invisible to managers Metrics make debt visible Measurement is the first step toward control Research Baseline Experimentation Trend analysis Quantify your experiments Make everything an experiment Make management decisions on measurements Create a dashboard Project Dashboard Rally the team Core metrics Issues Code size Complexity Team productivity Test coverage

Technology How does process affect leverage? Essential processes Technology Stack Deployment Quality and testing Continuous integration Versioning New skill acquisition Solve the meta-problem Tools matter Make vs. Buy Essential tools Natural lifespan of software Extend the life of an app Manage a series of products Build a business platform

Design - Creating a reusable architecture Creating a reusable architecture Strategic Interfaces Refactored frameworks Design patterns/idioms Evolutionary design

Code - Building reusable components Building reusable components Encapsulation & data hiding Extending functionality Testing scenarios Automation interfaces

Test - Testing that just works Testing that just works Beyond tactics Live data exercises Browser automation Diff testing