Net Objectives
WHITEPAPER

An Introduction to Commonality-Variability Analysis

by Al Shalloway and James R Trott

INTRODUCTION

Design patterns do not exist in isolation, but work in concert with other design patterns to help you create more robust applications. In this book, you will gain a solid understanding of twelve core design patterns and a pattern used in analysis. You will gain enough of a foundation that you will be able to read the design pattern literature, if you want to, and possibly discover patterns on your own. Most importantly, you will be better equipped to create flexible and complete software that is easier to maintain.

In this chapter, I will show how to use Commonality and Variability Analysis (CVA) to develop a high-level application design. Although design patterns can't be used in all designs, the lessons learned from them can. One of the most important of these lessons is that you can identify variation in your system using CVA. You can then follow the lessons of design patterns (program to interfaces, encapsulate variation using aggregation) to create designs that are flexible and easily testable.

COMMONALITY AND VARIABILITY ANALYSIS AND APPLICATION DESIGN

Isolating Variation is a Design Pattern Philosophy

Experienced developers know that when adding new function to an existing system, the major cost is often not in writing the new code, but in integrating it into the existing system. The reason for this is that the pieces in most existing systems are fairly tightly coupled. We must eliminate, or greatly limit, this coupling. One reason this occurs is that developers often consider how entities relate to each other before they are clear what the right entities are. From my experience of training people at all different levels of competency, I have come to the conclusion that more experienced developers do this even more than inexperienced developers. Developers need a way first to identify what they have before trying to find the relationships involved.

I suggest that you design applications in the following way: First, use CVA to identify the concepts (commonalities) and concrete implementations (variabilities) that are present in the problem domain. At this point we are mostly interested in identifying the concepts here, but many variabilities will be identified as part of this process. Any entities in the problem domain that are not included in these concepts (e.g., there may be some one-of-a-kind objects present) should also be identified. Next, once the concept for the functionality you need has been identified, you go on to specify the interface for the abstraction that encapsulates this. Derive this interface by considering how the concrete implementations derived from this abstraction will be used.

This approach basically follows Alexander's contextual design approach which is incorporated into the previously mentioned "Dependency Inversion Principle". By defining these interfaces, you are also determining which object use which objects – completing the specification of the design.

Let's see this in action with the CAD/CAM problem.

Solving the CAD/CAM Problem with CVA

Finding the concepts (and therefore abstract classes)

When analyzing the problem domain with CVA, I want to see what concepts are there and then try to organize these pieces as cohesively as possible. Remembering the CAD/CAM system, there are:

- Different CAD/CAM systems – V1, V2. In this situation, these are essentially read-only, proprietary databases that will provide the numerical control sets the expert system needs to do its work.
- Different kinds of Features – slots, holes, cutouts, special and irregular.
- Different kinds of Models – V1-based and V2-based.

To say there are different CAD/CAM systems really means that the concept is "CAD/CAM system" and the variations of it are "V1" and "V2".

From Chapter 15 of the Second Edition of *Design Patterns Explained: A New Perspective on Object-Oriented Design*. Shalloway and Trott. 2004.

CONTACT US
info@netobjectives.com
1.888.LEAN-244 (1.888.532.6244)

LEARN MORE
www.NetObjectives.com
portal.NetObjectives.com

Net Objectives

CVA leads to the following commonalities and their corresponding variations as shown in Figure 15-1.

Commonality	Commonality	Commonality
• CAD/CAM system	• Features	• Model
Variations	Variations:	Variations
• V1	• Slot	• V1-based
• V2	• Hole	• V2-based
	• Cutout	
	• Special	
	• Irregular	

An alternative approach is to pick any two items in the problem domain and ask the following questions:

1. Is one of these a variation of the other?
2. Are both of these a variation of something else?

For example, I might notice that there are features and slots. A slot is a kind of feature. I guess that "features" is a commonality and "slots" is a variation of it. Or, I might see that there are slots and holes. Within this problem domain, they do not seem to be variations of each other. They both seem to be variations of "features". Or I might compare the V1 CAD/CAM system and slots. There does not seem to be anything in common with them.

One issue per commonality

Of course, it is not always this simple. I may collapse concepts without realizing it. For example, suppose you think about the problem domain as having V1Slots, V1Holes, V2Slots, V2Holes, etc. The commonality would seem to be "CAD/CAM features". But this is commonality with two concepts: "CAD/CAM version" and "features". CVA says that commonalities should really be based on one issue per commonality. Otherwise I will not have strong cohesion in my design. Recognizing that I have two commonalities (CAD/CAM and Features) should lead me to ask which variations of these commonalities do I have? When I do this I come up with the variations I listed in Figure 15-1. This is one of the values of CVA: it results in cohesive concepts.

Relating the concepts

The next step involves determining how the concepts relate to each other. Models contain Features and Features are extracted from the CAD/CAM system. When I was designing this system, I thought it would be simpler if I built stand-alone Features that contained all of the information that had been in the CAD/CAM system that related to them. In other words, the CAD/CAM system was like a database to the Features – it contained the information about the Features. The Features presented the appropriate methods to make this information available, but the Features extracted this info from the CAD/CAM to get it.

The information in Figure 15-1 can be translated into three different class hierarchies by following the guidelines laid out in Figure 15-2. These are shown in Figure 15-3.

Models would also have to relate to the CAD/CAM system. Based on this analysis, the next level of detail is shown in Figure 15-4.

I am faced with a design decision here. Do I want to have different types of models or have one type of model that uses the different CAD/CAM systems through the CADCAM interface? In other words, if I move the methods in V1Model and V2Model that are peculiar to the CAD/CAM system into the CADCAM class hierarchy, I can probably avoid having different types of Models. This approach seems superior because Models use the CAD/CAM systems to implement

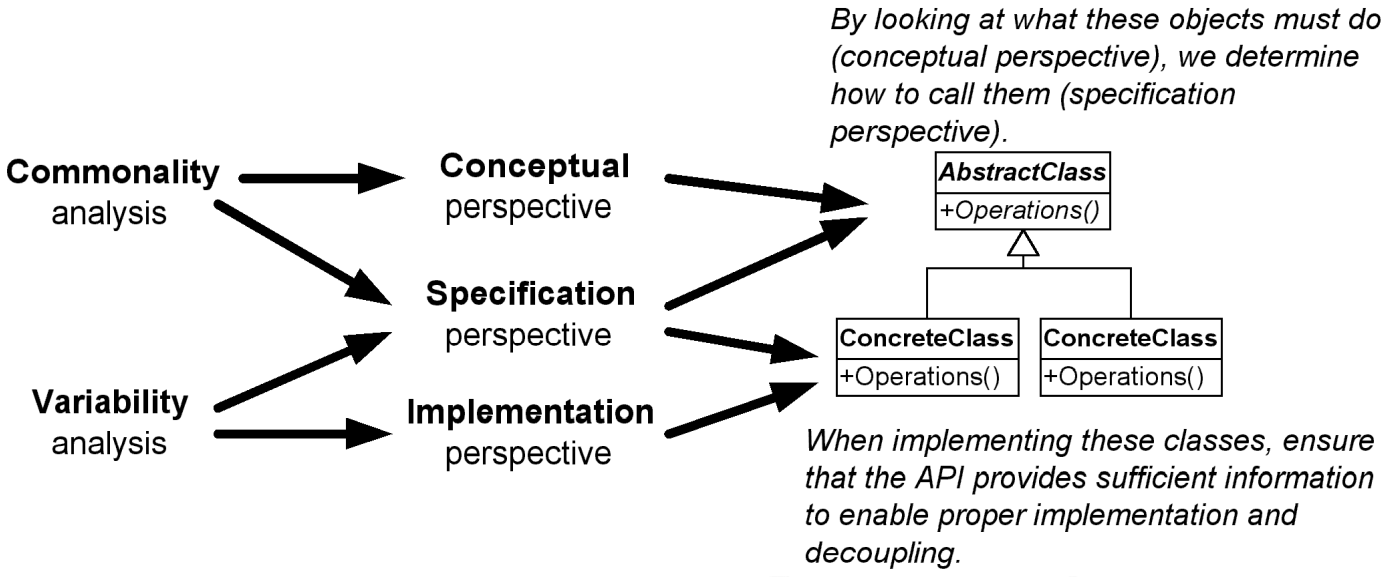


Figure 15-2 The relationship between commonality and variability analysis, perspectives, and abstract classes

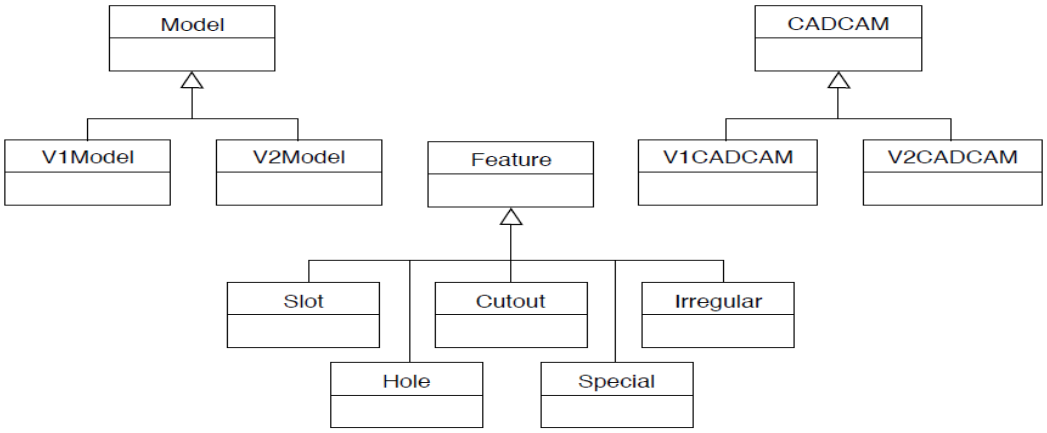


Figure 15-3 Translating CVA into classes

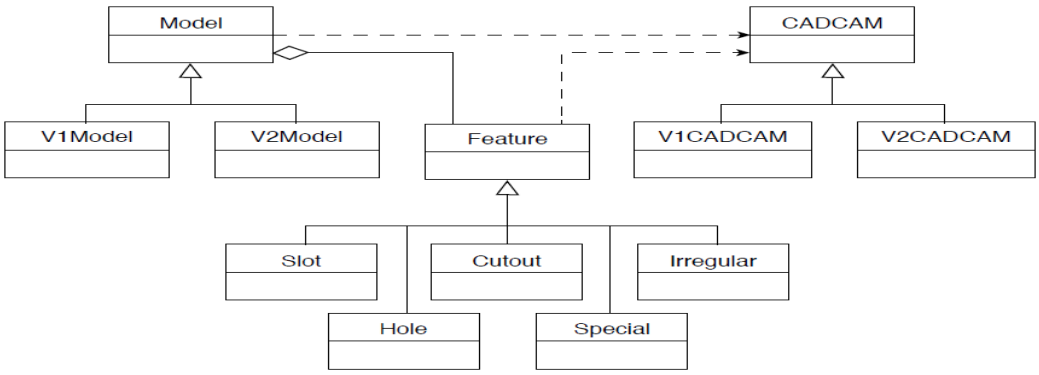


Figure 15-4 Our class diagram showing relationships between the classes

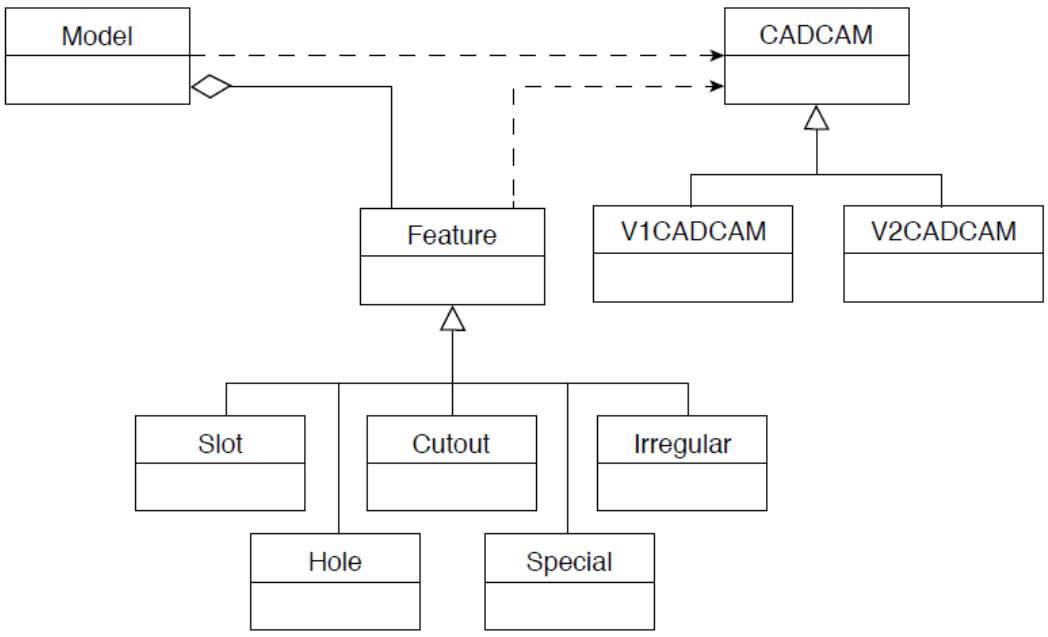


Figure 15-5 Handling variations in models by using the CADCAM classes

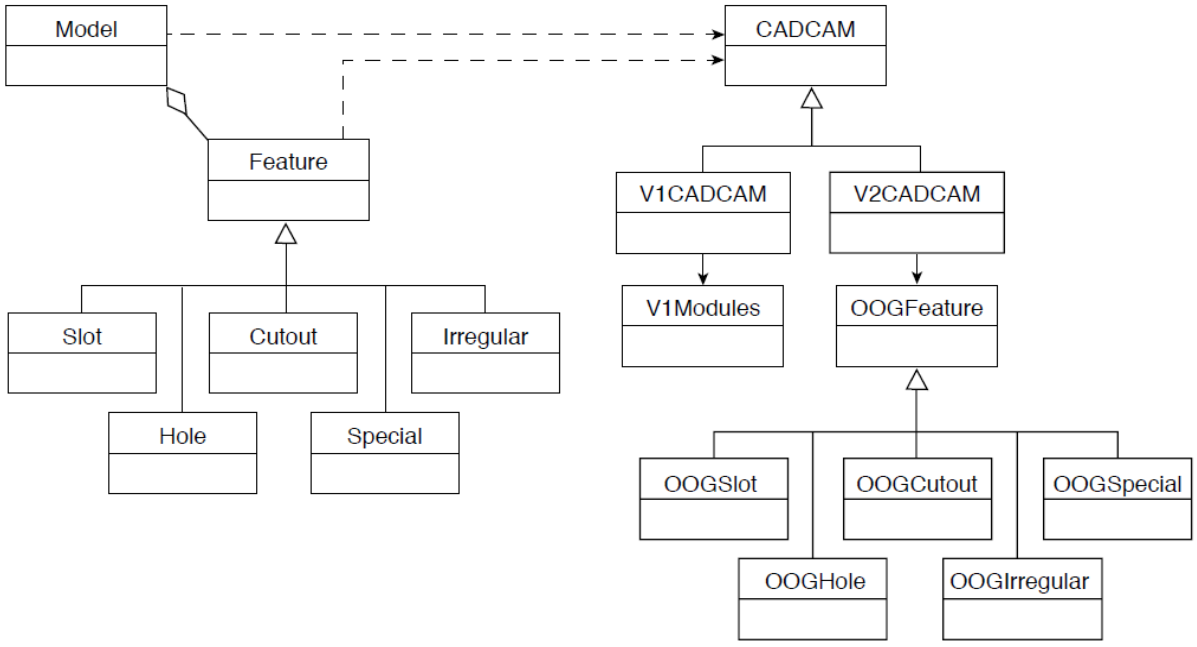


Figure 15-6 A completed design

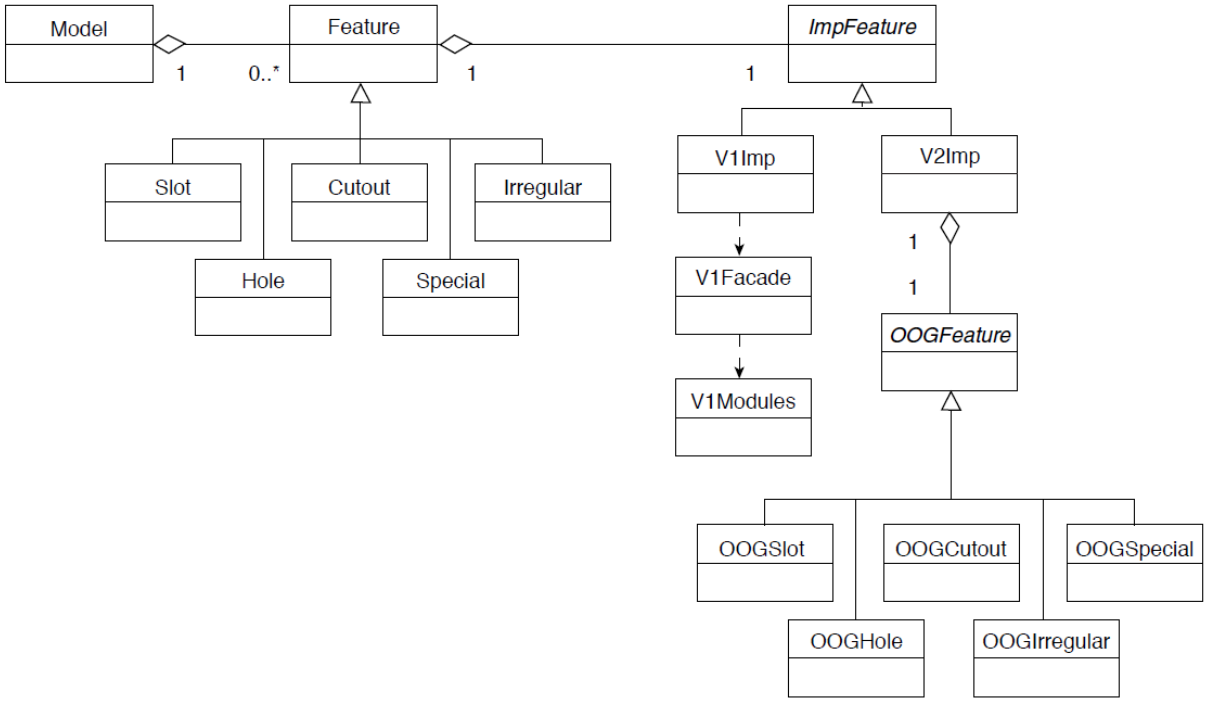


Figure 15-7 A solution using design patterns

them, but the concepts in Model are not inherently part of the CAD/CAM system. I show this approach in Figure 15-5. Don't get hung up on this distinction: there is not a great difference in code quality between the two of them as long as you make sure you do not have any redundancy. Personally, I like the solution shown in Figure 15-5 better because it has classes that are more cohesive. Therefore, I will use that for the rest of the design.

Expanding the design

I still need to expand the design to relate the CADCAM classes to the actual V1 and V2 implementations. Recalling what we know about the Facade and Adapter patterns, it should be clear that V1CADCAM should simply be a facade to the V1 system while V2CADCAM should wrap (adapt) the V2 system (OOG_Part). I show this in Figure 15-6.

The two look surprisingly similar. Or, maybe not.

The solution arrived at using design patterns was derived by using patterns in a contextual way. I applied one pattern at a time until the solution unfolded. This is very similar to the CVA approach:

1. Identify commonalities first.
2. Create abstractions from these.
3. Identify derivations with the variations of the commonalities.
4. See how the commonalities relate to each other.

This is another type of design by context. The interfaces of the classes are defined within the context of how they are used by other abstractions. The class definitions are similar in both approaches because CVA is just another way of finding what varies and encapsulating it in cohesive, loosely coupled classes – principles upon which the design patterns are based. The same principles and approaches therefore lead to remarkably similar solutions.

In truth, the two approaches are highly synergistic. CVA says to focus on abstractions early, which increases the probability that I will find the most useful ones. Design patterns focus on the relationships between those abstractions, but do not help identify those abstractions in the first place.

On the other hand design patterns allow me to apply specific insights from past successful designs, whereas CVA does not. For example, I know it is often desirable to keep Facades stateless due to the fact that they tend to be large, and so creating multiple instances can affect performance. Pulling the state needed by the features out of the Facade and placing it in the Adapter allows me to implement the Facade as a Singleton. CVA would not lead me to that conclusion.

SUMMARY

In this chapter, I explained how CVA can be used to create high-level application designs. By defining our commonalities first, we eliminate coupling between our special cases. Because design patterns are really about isolating variations and commonality and variability analysis, used the way I described, does the same thing, we can get similar solutions with CVA that we get using design patterns. The advantage of the CVA approach, however, is that it can be used all the time. We can only design with patterns when we know the patterns involved. Something that my experience shows doesn't happen that often.

REVIEW QUESTIONS

Observations

1. What are two approaches to identifying commonalities and variabilities?

Interpretations

1. CVA says you should have only one issue per commonality. Why is this important?
2. How do CVA and design patterns complement each other?

Opinions and Applications

1. "Experienced developers - even more than inexperienced ones - often focus on entity relationships too early, before they are clear what the right entities are". Is that your experience? Give an example to confirm or refute this statement.
2. Relate the approach to design starting with CVA with Alexander's approach.

Roadmap to Success

YOUR ROADMAP TO LEAN-AGILE SUCCESS

Growing numbers of organizations are realizing the need to become more Agile. Some are weighing the risks and benefits and seeking guidance. Others are implementing initiatives and are looking for ways to improve their return on investment.

The road to Lean-Agile success has become less risky as the early adopters have paved the way for the next generation of Lean-Agile methodologies and practices that solve the common problems, and transcend the limitations that early adopters have struggled with.

Net Objectives has been a thought leader in each of the Agile methods of the past decade. This uniquely enables us to provide the most effective approach to our clients' needs.

For more than a decade, Net Objectives has been training and facilitating large and small organizations to achieve agility.

We serve organizations at the team, management, and enterprise level with comprehensive organizational consulting, coaching, and training.

We do not promote one method as most other firms do – rather we pull from a broad knowledge base to offer an approach tailored to your situation.

UNDERTAKING YOUR TRANSITION TO AGILE

There is no one method that guarantees success at the team level. Our full assessment services will answer these questions to help you to determine which to choose.

- Do cross functional teams already exist and if not, how difficult will it be to create them?
- Are certain staff essential for multiple teams
- How many concurrent projects are teams working on at one time?
- What challenges face the organization in integration and deployment?

Throughout your transition, Net Objectives will help ensure everything is in place through appropriate Lean-Agile training:

- **Teams** are capable of delivering value quickly with high value
- **Businesses** are capable of selecting, sizing and prioritizing business capabilities to be developed
- **Management** takes responsibility for improving the value stream and removing impediments facing teams

Our coaches enable your teams with skills, and competencies to leverage the power of agility as part of your value stream. Our consultants collaborate with management, stakeholders, executives, and experts to provide insight and guidance from the organizational view.

UNDERSTANDING AGILE

The first step toward success is drawing a clear distinction between enterprise agility, and team agility. The benefits of Agile at the team level are very different than benefits at the enterprise level. The paths to success and the challenges presented are also very different.

Enterprise agility enables an organization to effectively respond at the enterprise level to changing business needs while reliably delivering business value.

Team agility is a component of that capability – a component – and not the equivalent of enterprise agility. This understanding is essential since team methods alone cannot deliver enterprise level benefits.

First generation methods made the assumption that team agility translated to the enterprise. This has been a costly simplification.

Many organizations have attempted to achieve enterprise agility simply by creating more Agile teams. This often starts well, but usually ends up being impeded by enterprise level problems that team solutions do not solve.

The next generation of Lean-Agile openly acknowledges practical truths, limitations, and organizational structures required to fulfill the needs of the entire Lean -Agile enterprise.

From assessment and planning to pilot and rollout, our goal is to facilitate your organization with custom approaches and solutions that are appropriate to your needs, structure, and goals. Let us show how the next generation of Agile can benefit your organization.

info@netobjectives.com www.NetObjectives.com 1.888.LEAN-244 (1.888.532.6244)

Drive from Business Value

BUSINESS-DRIVEN SOFTWARE DEVELOPMENT

Business-Driven Software Development is Net Objectives' proprietary integration of Lean-Thinking with Agile methods across the business, management and development teams to maximize the value delivered from a software development organization. This approach has a consistent track record of delivering higher quality products faster and with lower cost than other methods.

Business-Driven Software Development goes beyond the first generation of Agile methods such as Scrum and XP by viewing the entire value stream of development. Lean-Thinking enables product portfolio management, release planning and critical metrics to create a top-down vision while still promoting a bottom-up implementation.

Our approach integrates business, management and teams. Popular Agile methods, such as Scrum, tend to isolate teams from the business side and seem to have forgotten management's role altogether. These are critical aspects of all successful organizations. Here are some key elements:

- Business provides the vision and direction; properly selecting, sizing and prioritizing those products and enhancements that will maximize your investment.
- Teams self-organize and do the work; consistently delivering value quickly while reducing the risk of developing what is not needed.
- Management bridges the two; providing the right environment for successful development by creating an organizational structure that removes impediments to the production of value. This increases productivity, lowers cost and improves quality.

BECOME A LEAN-AGILE ENTERPRISE

Involve all levels. All levels of your organization will experience impacts and require change management. We help prepare executive, mid-management and the front-line with the competencies required to successfully change the culture to a Lean-Agile enterprise.

Prioritization is only half the problem. Learn how to both prioritize and size your initiatives to enable your teams to implement them quickly.

Learn to come from business need not just system capability. There is a disconnect between the business side and development side in many organizations. Learn how BDSD can bridge this gap by providing the practices for managing the flow of work.

WHY NET OBJECTIVES

While many organizations are having success with Agile methods, many more are not. Much of this is due to organizations either starting in the wrong place, such as focusing on the team when that is not the main problem, or using the wrong method, such as using Scrum or kanban because they are popular.

Net Objectives is experienced in all of the Agile team methods (Scrum, XP, Kanban) and integrates business, management and teams. This lets us help you select the right method for you.

LEARN TO DRIVE DEVELOPMENT FROM THE DELIVERY OF BUSINESS VALUE

What really matters to any organization? The delivery of value to customers. Most development organizations, both large and small, are not organized to optimize the delivery of value. By focusing the system within which your people are working and by aligning your people by giving them clear visibility into the value they are creating, any development organization can deliver far more value, lower friction, and do it with fewer acts of self-destructive heroism on the part of the teams.

THE NET OBJECTIVES TRANSFORMATION MODEL

Our approach is to start where you are and then set out a roadmap to get you to where you want to be, with concrete actionable steps to make immediate progress at a rate your people and organization can absorb. We do this by guiding executive leadership, middle management, and the teams at the working surface. The coordination of all three is required to make change that will stick.

OUR EXPERTS

Net Objectives' consultants are actually a team. Some are well known thought leaders. Most of them are authors. All of them are contributors to our approach.

Al Shalloway

Scott Bain

Luniel de Beer

SELECTED COURSES

Executive Leadership and Management

Lean-Agile Executive Briefing

Preparing Leadership for a Lean-Agile Transformation

Product Manager and Product Owner

Lean-Agile Product Management

Lean-Agile at the Team

Acceptance Test-Driven Development

Implementing Team Agility

Team Agility Coaching Certification

Lean-Agile Story Writing with Tests

DevOps

DevOps for Leaders and Managers

Train the Trainer

Online Coaching Academy

Becoming a Team Agility Trainer

Technical Agility

Advanced Software Design

Design Patterns Lab

Effective Object-Oriented Analysis and Design

Emergent Design

Foundations of Sustainable Design

Sustainable Test-Driven Development

SAFe®-Related

Leading SAFe® 4.0

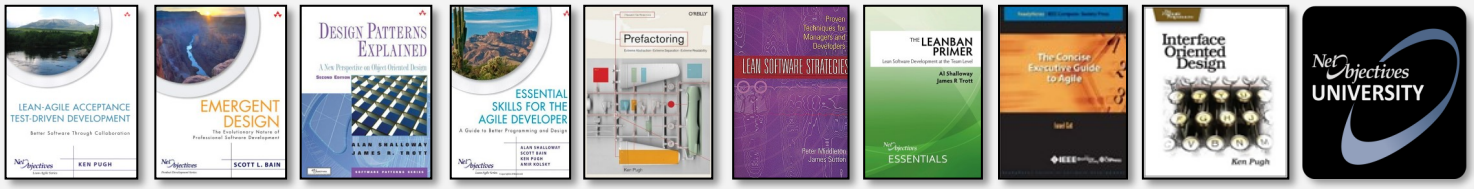
Using ATDD/BDD in the Agile Release Train (workshop)

Architecting in a SAFe Environment

Implement the Built-in Quality of SAFe

Taking Agile at Scale to the Next Level

OUR BOOKS AND RESOURCES



CONTACT US
info@netobjectives.com
1.888.LEAN-244 (1.888.532.6244)

LEARN MORE
www.NetObjectives.com
portal.NetObjectives.com

Net Objectives