Agile Development with Agile Business Suite

By: Alan Hood
Agility. It’s a wonderful thing.

For a dancer or an athlete the need for agility is obvious. The ability to move, and change directions quickly, and with little apparent effort, is a highly prized quality.

In business agility means being able to react to changing requirements quickly and efficiently. To lead in your market, not just follow along after everyone else has beaten you to the punch.

For applications this means being able to take the needs and wants of your clients and business leaders into account, and then when their needs change, change with them.

A style of application development, called Agile Software Development has evolved to address the need for greater agility in this area.

With Agile Business Suite (now, AB Suite™) you have this ability built in to the product.

Believe it or not, the word “Agile” in the name of Agile Business Suite was not originally intended to refer to the “Agile” development methodologies, but the product is extremely easy to use in that environment.

So let’s look at how to use AB Suite in an Agile development environment.
# Table of Contents

**Agile Concepts and Agile Development**  
- Key Concepts  
  - Iterative, Incremental, Evolutionary  
  - Small Teams, with Efficient, Face-to-Face Communication  
  - Very Short Feedback and Adaptation Cycle  
- The Agile Manifesto  
- Some Popular Agile Methods  
- Scrum  
  - Organization of a Scrum Team  
  - What Makes a Good Story?  
- Comparison of Different Methods  
  - Waterfall Approach  
  - Iterative Approach  
**Being Agile with AB Suite**  
- AB Suite Overview  
- AB Suite and the Agile Manifesto  
- Additional Features  
**Closing Remarks**  
**About the Author**

---

**Not familiar with AB Suite? Would you like more information about it?**

Go to www.unisys.com for a description, links to our electronic newsletter – Developing Agility™, Product Information and software downloads, client case studies, and more.

Unisys can also provide technical services with AB Suite, and installing and using Microsoft Visual Studio and Team Foundation Server. Contact ABSuite@Unisys.com for more information.
Agile Concepts and Agile Development

The concepts that make up agile software development have been developed over many years. In 2001 a group of industry experts met to find some common ground in many of the successful development methodologies that were being promoted at the time. The result was a Manifesto for Agile Software Development, which has more or less become the guiding principles for all “Agile” development tools, processes, and schools of thought since that time. A summary of the twelve principles is listed in this paper, and the full document, along with a brief history and other information can be found on the organization’s web site. A link is provided below.

Key Concepts

Iterative, Incremental, Evolutionary

In order to be agile, an organization must be able to provide updates to the software very quickly. You can’t make users wait for months or years for the features they have demanded and call yourself agile. This principle is demonstrated through delivery of frequent, small, but complete working samples. This is sometimes known as Iterative or Incremental Delivery, or Evolutionary Prototyping.

Small Teams, with Efficient, Face-to-Face Communication

In order to ensure effective communication between all of the team members it is necessary to have frequent, focused meetings to provide updates and status to one another. Ideally this should be done with a very small team of developers, and ideally all in the same place, at the same time.

At the time the Manifesto was written, in 2001, the only way to ensure ongoing and efficient communication was if the team was co-located and could all meet together every day. Today with advances in business oriented social media and electronic communications the need for everyone to be in the same physical location is reduced, but there is still a very strong desire for effective communications between all members of the team.

Very Short Feedback and Adaptation Cycle

With the pace of incremental deliveries to users increasing, it is also necessary to receive timely feedback from the users and business stakeholders, and then incorporate that feedback into the product as quickly as possible. In an ideal environment the first level users should be able to provide feedback on a given increment before the next one is available, and depending on the project priorities, changes resulting from high priority bugs or requirements should be included in the product within one or two increments.

There is a continuous focus on improving the quality of the product, and the efficiency of the team.

The Agile Manifesto

Paraphrased from: http://www.agilemanifesto.org/principles.html

- Customer satisfaction through early and continuous delivery of valuable software
- Welcome changing requirements, even late in development
- Deliver working software frequently (weeks rather than months)
- Business people and developers work together throughout the project (daily)
- Build projects around motivated individuals, who should be trusted
- Face-to-face conversation is the most efficient form of communication (co-location)
- Working software is the primary measure of progress
- Sustainable development, able to maintain a constant pace
- Continuous attention to technical excellence and good design
- Simplicity—the art of maximizing the amount of work not done—is essential
- Self-organizing teams
- The team regularly reflects on how to become more efficient, and adjusts accordingly
Some Popular Agile Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive Software Development (ASD)</td>
<td>Agile modeling</td>
</tr>
<tr>
<td>Agile Unified Process (AUP) – a simplified RUP</td>
<td>Dynamic systems development method (DSDM)</td>
</tr>
<tr>
<td>Extreme Programming (XP)</td>
<td>Feature-driven development (FDD)</td>
</tr>
<tr>
<td>Kanban</td>
<td>Lean software development</td>
</tr>
<tr>
<td>Scrum</td>
<td></td>
</tr>
</tbody>
</table>

We don’t actually have a single process that could be described as “the Agile Methodology”. There are probably more than a dozen methodologies or frameworks that have been defined in support of Agile development, with many variations of each. The table above lists some of the most popular ones. They all have in common the concepts of rapid, frequent releases, continuous improvement, and effective communications with the customers.

Other popular methods that emulate Agile include DevOps (Development Operations) and RAD (Rapid Application Development). While they embrace many of the values of Agile development, and have many of the same benefits, they are not generally considered “Agile development”.

Scrum

Scrum is probably the most popular Agile development method. It is characterized by small, mostly co-located, self-managed teams. Software is delivered in multiple, brief iterations, called Sprints. Requirements and tasks are defined as Stories.

This symbol, from the Scrum Alliance, represents the Scrum cycle. The lower circle represents the Sprint. Stories come in from the left, are completed during the Sprint, and exit on the right as “done”. The upper circle represents the daily Scrum Meeting in which each member of the Scrum Team reports on what they have done since the last meeting, what they plan to do next, and any impediments they are experiencing. The output from each Sprint should be working software that implements each of the Stories completed during the Sprint.

The Sprint Backlog is the set of Stories the team has agreed to complete during the current Sprint. Each Sprint begins with Sprint planning, in which the team will work through the Product backlog to define the work items for this Sprint. At the end of each Sprint there will be demonstrations for the stakeholders, and the team will conduct a Retrospective to describe what went well, and where they can improve their processes in the near future.

Organization of a Scrum Team

A Scrum Team is small, usually consisting of 5-8 members, including a Scrum Master, a Product Owner, and Team Members.

Typical Scrum Team

- **Scrum Master** – leads the Scrum, not as a boss or manager, but more as a facilitator. In some organizations there are people with the designated role of Scrum Master. In others the team will take turns acting as Scrum Master.
- **Product Owner** – speaks for business users (stakeholders) in the scrum. The Product Owner is the member of the team with the final say on what the team is building, not how it is being built. If there are questions about the intent of a Story, or how it might impact the product, the Product Owner will make the decisions, and/or communicate with the external stakeholders about it.
• Team members – developers, testers, subject matter experts as needed. There are usually 3-6 developers who round out the team.

• Business Owner and Stakeholders - not part of the core team. In larger organizations there may be a Business Owner and a group of people who represent the stakeholders. These people define the requirements, which means they write the user Stories that will define the product. At the end of each Sprint they receive the product and provide feedback to the Scrum for continuous improvement. In general these people will primarily interact with the Product Owner, not with the rest of the Scrum Team.

What Makes a Good Story?
In Scrum requirements and features are all defined as user “Stories”. A story should be small enough that it can be completed by the Scrum in one Sprint. A good story should describe who needs or wants it, what they want to do, why they need it, and how they will know it is done. This last component is critical, and probably the part that is most often forgotten, or poorly stated. It is essential that everyone agree on what constitutes “done” for a story.

During the Sprint planning the Scrum Team will identify the internal tasks they will need to accomplish in order to complete the Story, and each member will select their tasks from the prioritized Scrum backlog.

If a Story cannot be completed within a Sprint it will usually go back to the product backlog for the next Sprint. In that case the Product Owner will need to communicate with the business stakeholders to be sure there are no surprises when they receive the next increment of the product.

If a Story is too large to be completed in one Sprint, or if it must be broken up to be worked on by more than one Scrum team, that Story will become an Epic, and two or more appropriately sized user stories will be written to replace it.

Examples
This is an example of an Epic Story that might take a team more than one Sprint to complete.

“As a mobile phone user, I want secure iPhone access to my ClearPath banking transactions, so that I can make deposits, transfers, and check account balances. I’ll know it’s done when I can do all of my online banking from my iPhone.”

One of the User Stories that might come from that is this.

“As an iPhone user, I want secure Web access to the account deposit transactions, so that I can make deposits into my checking account. I’ll know it’s done when the same functionality on the DEPST transaction is available on iPhone.”

This User Story is a good example of one that could easily be accomplished by a development team using AB Suite and ClearPath ePortal to provide a new mobile interface for an existing application.

Breaking it down into its component parts –

<table>
<thead>
<tr>
<th>As an iPhone user</th>
<th>Who is requesting the feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>I want secure Web access to the account deposit transactions</td>
<td>What they require</td>
</tr>
<tr>
<td>so that I can make deposits into my checking account</td>
<td>Why they need it</td>
</tr>
<tr>
<td>I’ll know it’s done when the same functionality on the DEPST transaction is available on iPhone.</td>
<td>How they will know the feature is done</td>
</tr>
</tbody>
</table>

In some organizations, especially larger ones, or with larger products, two Story layers – Epics and User Stories – may not be enough. It may be necessary to define a set of Epics, as say Feature Stories, and possibly even a collection of Features as Release Stories. This is not part of the definition of “pure” Scrum, but it is one way in which some larger organizations have adapted the concepts of Scrum to fit their needs.

For more information about Agile and Scrum you can visit these web sites.
www.agilemanifesto.org the home of the “Twelve Principles of Agile Software”
https://www.scrumalliance.org the Scrum Alliance is an organization that fosters education and enhancement of the Scrum framework
Comparison of Different Methods

Waterfall Approach

Traditional application development follows what can be described as a Waterfall approach. This is a multi-step process in which the requirements are heavily front loaded, and designs (and documents) to satisfy those requirements are developed and agreed upon with the stakeholders in great detail before any coding is started. Many organizations use this process, or one very much like it. The labels for each step may change, but they typically have six to eight, or even twelve steps from initial inception and requirements definition until the final deployment of the finished product. Typically with this model the users may be involved at the very beginning and again at the end, but have very little contact with the development team throughout the rest of the project.

One of the greatest problems with a traditional, “waterfall” approach is that requirements are fixed very early in the project, and even if the development team is 100% effective in meeting those requirements, by the time the product is delivered, perhaps several months later, the user’s real requirements have already changed significantly. So on the day the new product goes live there is already a pent up demand for new functionality. We can call the difference between what is wanted by the users and what is delivered by the product a Requirements Gap.

Iterative Approach

With an iterative approach the work is done in smaller pieces. There is still a need for a “big picture” view of what functionality the product will eventually provide, but the user community can be exposed to the product at several points during development. Requirements can be reviewed and refined, and the product can become more complete with each iteration.

Requirements vs. Capabilities Delivered

With this model the users may be involved at the very beginning and again at the end, but have very little contact with the development team throughout the rest of the project.
A more iterative approach combines early feedback from the users with the ability to change the product to respond to changing requirements. This means that when the product is delivered and ready to go into production the Requirements Gap between what the product provides and what the users really want can be much less significant.

**Being Agile with AB Suite**

Unisys has never taken the approach of prescribing any particular development methodology with AB Suite. Instead we have encouraged clients to use the product in a way that works best for them. It is very flexible, and can be adapted to work with many different styles. However, having said that, an iterative development process that has many of the hallmarks of Agile has always been at the heart of AB Suite and its predecessors. Some of our most effective, and most successful users have employed an “agile” approach, even before the term became popular.

**AB Suite Overview**

Unisys AB Suite is a unique Model Driven environment for developing mission critical solutions. It supports design, development, generation, and ongoing maintenance of high-performance, highly available, enterprise-class applications. Users develop their applications in an AB Suite model and then AB Suite will generate the target application, including transactions, batch programs, database, and even GUI clients, all from the model. And being model driven, when they want to make changes to the application they just change the model and regenerate the application. AB Suite takes care of building and deploying the parts that have changed.

AB Suite provides a development environment in which business and technical professionals truly collaborate, directing changes and designing new capabilities at the business level. It encourages people to focus on what an application should do and not worry about how it should be coded or deployed. AB Suite is composed of Developer and Runtime for each of the two supported platforms, ClearPath® MCP and Windows.

When installed as a package in Visual Studio, AB Suite takes center stage in Application Lifecycle Management. Features in Visual Studio and Team Foundation Server that foster agile software development are as valuable, or even better, when AB Suite is selected as the development language of choice.

AB Suite Developer works as a package within Microsoft Visual Studio. This makes it easy to learn, and allows users to take advantage of the power of the extended capabilities in Visual Studio Team Foundation Server.

Within AB Suite Developer, the three highly-integrated modules are:

- The System Modeler, which is the primary design and development tool
- A Debugger for executing unit and functional tests
- A Builder to automatically generate and deploy the application AB Suite Runtime provides the framework in which applications run.
AB Suite and the Agile Manifesto

Let’s look at the summary points of the Agile Manifesto and see how easily AB Suite can be used to accomplish these lofty goals.

Customer satisfaction through early and continuous delivery of valuable software

The model driven approach in AB Suite, combined with complete generation of the application in the target environment makes it incredibly easy to prototype new applications, or changes to existing AB Suite applications. You can test the changes directly in the development environment without building the application. Demonstrate the new functionality to your users, and build it for test or production, knowing that what you demonstrated and tested is what you will deliver.

Welcome changing requirements, even late in development

Here the key point is the 100% generation of the target applications. Since the developers don’t need to worry about how the application must be coded for the target environment, they can concentrate on implementing the business requirements (“Stories” – remember?) in the model, using the high level, platform independent scripting language in AB Suite. Test changes and new functionality with the Debugger and ATT (Automated Test Tool), and quickly provide an updated version for the users to try out.

Deliver working software frequently (weeks rather than months)

The Build process in AB Suite is optimized to detect changes in the model and only build those parts of the application that are necessary. If you have an application with hundreds of transactions and batch programs, and you make a change that effects only a small number of them, Builder will determine which of the objects have been impacted by your changes, and build, compile, and deploy just those that have changed. This process actually encourages frequent incremental deliveries, because it is so easy to build a complete application that incorporates just your critical changes.

Business people and developers work together throughout the project (daily)

The model driven, team centered nature of AB Suite makes is easier for users and technicians to work together. They can find common ground in the AB Suite business model. It does not include the lower level details required to deploy an application; things like database schemas or business rules that have been translated into lower level programming languages such as COBOL or C#. The programmer can prototype interfaces, like graphical screens or mobile presentations in AB Suite, and show the changes to the business user, right in Developer. When they agree they can build and test the application. If it isn’t quite right they can go back to Developer to make minor changes, and then build and test it again.

Not only does this make the development process more interactive and responsive, it also encourages the developers to learn more about the business, which helps improve communications all around.

Build projects around motivated individuals, who should be trusted

Studies have shown that when people feel they have more involvement in what they will do and how they will do it, they are more highly motivated, Since AB Suite has a tendency to bring the business people and the developers closer together, they develop a better understanding of the business, and the reasons behind features and functionality they are requesting. AB Suite allows small teams to work together to build and maintain very large, complex application systems. They have the freedom to concentrate more on satisfying the users’ needs, and worry less about the low level details of how the applications will be built and deployed.

Some of the most dedicated individuals are AB Suite developers. They just really like what they do.

Face-to-face conversation is the most efficient form of communication (co-location)

As stated earlier, AB Suite works as a package in Visual Studio. When your users and developers are all in one place, in a daily Scrum meeting, they can use the project tracking features in Visual Studio to report on progress. But even if they can’t all be in one place at one time, the same project tracking features in Team Foundation Server can be used to keep everyone up to date on the latest news and project status.

Other tools, such as Lync or Skype, project portals, and good old fashioned email, can also help to keep everyone “on the same page”, whether they are located in the same office, or in different time zones.
**Working software is the primary measure of progress**

With AB Suite the model is the application. If the model, as defined and demonstrated in AB Suite Developer satisfies the needs of the business and end-users, that is exactly what will be deployed the next time you choose to Build it.

AB Suite delivers a fully working system right from the very beginning, even if the developer has only defined a segment and a single Ispec class.

**Sustainable development, able to maintain a constant pace**

In AB Suite we have a saying, “Always complete, never finished.” What that means is that the model of the application in AB Suite will always result in a complete, working application when it is deployed – to the extent you have defined it. But in a world where requirements are constantly changing, being able to make changes in the model – whether they are large or small – and have confidence that you can easily understand the impact of these changes, makes continuous development not only possible, but highly effective.

**Continuous attention to technical excellence and good design**

With AB Suite you have several ways to look at an application model. You can explore the Class View, display a UML diagram of all or parts of the application, use change tracking to identify what modifications have been made over time, or in a specific Sprint or release. Fast Find and Cross Referencing tools help you identify and isolate the scope of changes and perform impact analysis before you make a change.

Since AB Suite generates the full application, you don’t need to worry about the low level technical details. When new features are available in the target platform, such as relaxing of database limits, a new version of the operating system, or introduction of a new .NET Framework, you get the benefit of these changes automatically, because an updated version of AB Suite will be able to build for that environment, often without requiring any changes in your AB Suite model.

**Simplicity—the art of maximizing the amount of work not done—is essential**

AB Suite raises the level of abstraction by at least an order of magnitude over traditional development environments or languages like COBOL or C++/C#. Developers write less code. They can concentrate more on what the application should do, and less on how it should do it. And when the time comes to make a change, they use exactly the same environment to make the change.

**Self-organizing teams**

Many traditional organizations have a deep hierarchy of development professionals. There may be architects, business analysts, system analysts, designers, programmers, testers, database administrators, user interface specialists, and much more. Each is a specialist, and there is little crossover between the disciplines. A typical AB Suite development team is organized much more like a Scrum team. There is frequently an architect (the Program Owner) usually a project manager (the Scrum Master) and a small number of developers (the Scrum Team). Since AB Suite automatically generates “the hard stuff”, like the database schema, standard job control, and even many types of clients and programmatic interfaces, there usually is no need for specialized technical skills in the team. And when they are needed, they can be brought in as SMEs on a temporary basis.

**The team regularly reflects on how to become more efficient, and adjusts accordingly**

The key here is communication, and a willingness to look back on what you have done, and find ways to do it better the next time.

Unisys is constantly publishing “best practices” for various aspects of the AB Suite environment, as well as a quarterly newsletter that includes articles on various technical points of the product. There are more than 100 “how to” articles on a wide range of topics on the AB Suite support web site. These are completely free of charge to any user. And of course there are white papers like this one that go into a bit more detail in areas that people may have an interest.

**Additional Features**

There are many other features of AB Suite that foster agile development, apart from the practices and features described above that directly address some aspect of Agile Manifesto. These include the following.

**Shared Repository**

The AB Suite business model is stored in a shared repository using Microsoft SQL Server. Developers can share the repository, and thereby have access to changes made by any member of the team.
AB Suite also supports version control and release management using Team Foundation Server. Even very small projects can receive benefits from this level of management and control. Larger projects benefit even more, because the multi-user repository allows them to keep track of several versions, or iterations of the application. They can easily stage releases from the output of each Scrum’s activities. And if there happen to be problems, or conflicting changes when they bring it all together, it is easy to determine what caused the problem and back out those changes.

**Define Presentation for any object**

For any object that is potentially visible to the outside world, whether that is a screen, a report layout, a single data item, or a class that defines the structure of a mailing address, you have the option of defining its presentation properties right along with the other properties of the item. This makes the usage of these objects consistent throughout the application. It is also a tremendous time saver when you design a form or transaction using the Form Painter. Drag that address class onto the palette and you will automatically inherit the presentation formatting that has been defined for it.

**Dictionary**

AB Suite maintains a built-in dictionary of any data items or other objects you define in the application model. You can define the properties of an item (name, type, size, usage, even presentation) in the dictionary, and then when you want to use one of those items in a class or presentation, simply select it from the dictionary, or inherit properties from an item in the dictionary. This will not only make it faster and easier to create your business objects the first time, if later on the definition of the item is changed the modifications will automatically be inherited throughout the application model.

**Documentation**

For some people “Documentation” is a bad word, but it doesn’t have to be. Every object in the AB Suite model, from the smallest data item, to the Solution container itself, has documentation property. Developers can use this to describe how the object should be used, explain complex algorithms, define user interfaces, and even document parts of the design that have not been completed yet. This documentation can be extracted and printed separately, making production of design documents or user manuals that much easier. And since the documentation is part of the object it is easier for developers to keep it up to date when something changes in the future.

**Debugger – test without building**

Unlike most development products, it isn’t usually necessary to build and compile an AB Suite application in order to test it. The developers or administration staff can define Debug as well as Release Configurations for a project, and the developers or testers can run the application in debug mode right in Visual Studio. Most application objects will be interpreted directly from the repository without the need to be compiled. Developers can set tracing and breakpoints in the logic, step through code, examine and modify data. They can make code changes, and step through the changes again – all without the delays of building the entire application.

Some programs, such as screen interfaces, may need to be compiled before they can be tested, but AB Suite takes care of that also, by doing a “Just in Time” build of only those programs that need it.

This makes prototyping easier. It also makes it easier to demonstrate your new functionality to stakeholders at the end of a Sprint. If they have a question about something you have done, jump into the Debugger and show them.

**Integration with Team Foundation Server**

When used with Visual Studio, AB Suite Developer is also well integrated with Team Foundation Server (TFS). It is not required for users to install and use TFS with AB Suite, but they may see significant benefits in project management and control, and improved team communications if they do. Version Control with TFS has been a feature in AB Suite for years. More recent releases have seen integration with TFS Test Manager, Build Manager, Release Manager, and requirements and project tracking features of Team Foundation Server.

Since AB Suite works with TFS, you can use various Agile process templates and reports that help you to manage your Stories, Sprints, Backlogs and more with AB Suite in the same familiar environment. For example, the Scrum Process template can be found at this location in the MSDN library - https://msdn.microsoft.com/en-us/library/ff731587.aspx

(Note, this is an external link to the Microsoft Developers Network website, and it may change, or may require a subscription.)
**ATT and Team Test**

ATT (Automated Test Tool) is a feature in AB Suite that allows developers or testers to define and execute test cases for transactions simply by running the transactions. These test cases can then be stored and rerun any time using the Team Foundation Server Test Manager.

Automated testing makes it easier to run the tests, which means they will actually be run. It also helps ensure you haven’t introduced any regressions in a part of the application you didn’t intend to change.

**Change once, change everywhere**

Since AB Suite is model driven, repository based, and generates the full application, it is amazingly easy to make changes. And if you make a change in one area, any part of the model that uses that object is automatically validated and rebuilt if necessary.

As an example, let’s say you originally defined your application to use telephone numbers that were 7 numerical digits. Then you had to add a three digit area code, because the phone companies now started using 10 digits numbers. If you change the definition of the telephone number in the AB Suite dictionary, it will find all of the places the phone number is used, and any objects that inherit from the phone number, and change them automatically. This will include the internal representation, any database items, report outputs, and transactions.

Now go one step further, and assume your company has been so outrageously successful that you are now starting to have customers in other countries. You need to change your address objects to include country names, longer postal codes, and now it seems like a good idea to store the telephone numbers as 16 character alphanumeric data. No problem. Change the dictionary definitions for those classes, test it, build it, and go. If there is a problem with any of your changes – an incompatibility in the usage, or a change in type or length that might cause a long, expensive database reorganization, for example – AB Suite has the tools to help you find those places, do the impact analysis to know what will need to change before you change it, and find and fix any incompatibilities before you release it.

That’s pretty agile. Don’t you agree?

**Closing Remarks**

Whether you have chosen to use a traditional application development methodology, DevOps, RAD, Agile, Scrum, or some hybrid combination of your own, AB Suite is a development environment that can be used effectively, and increase your chances of success. From the very beginning AB Suite has been designed to make developers more productive, to bring business people and developers closer together, and to make it easier to adapt and respond to changing business needs. In a nutshell this is the very definition of an agile organization.

If you would like more information about AB Suite, check out our web site at [http://www.Unisys.com](http://www.Unisys.com), or write to us at ABSuite@Unisys.com

**About the Author**

As a consulting engineer, architect, and senior technical consultant, Alan Hood has been helping people get the most from their application development tools and processes for years. He works with clients across the globe to develop and enhance their core business applications. Alan worked for many years in leadership engineering and technical services roles for Unisys, and is now an independent consultant. He is a certified Scrum master, and has been using, teaching, and coaching agile development methodologies for more than 15 years.