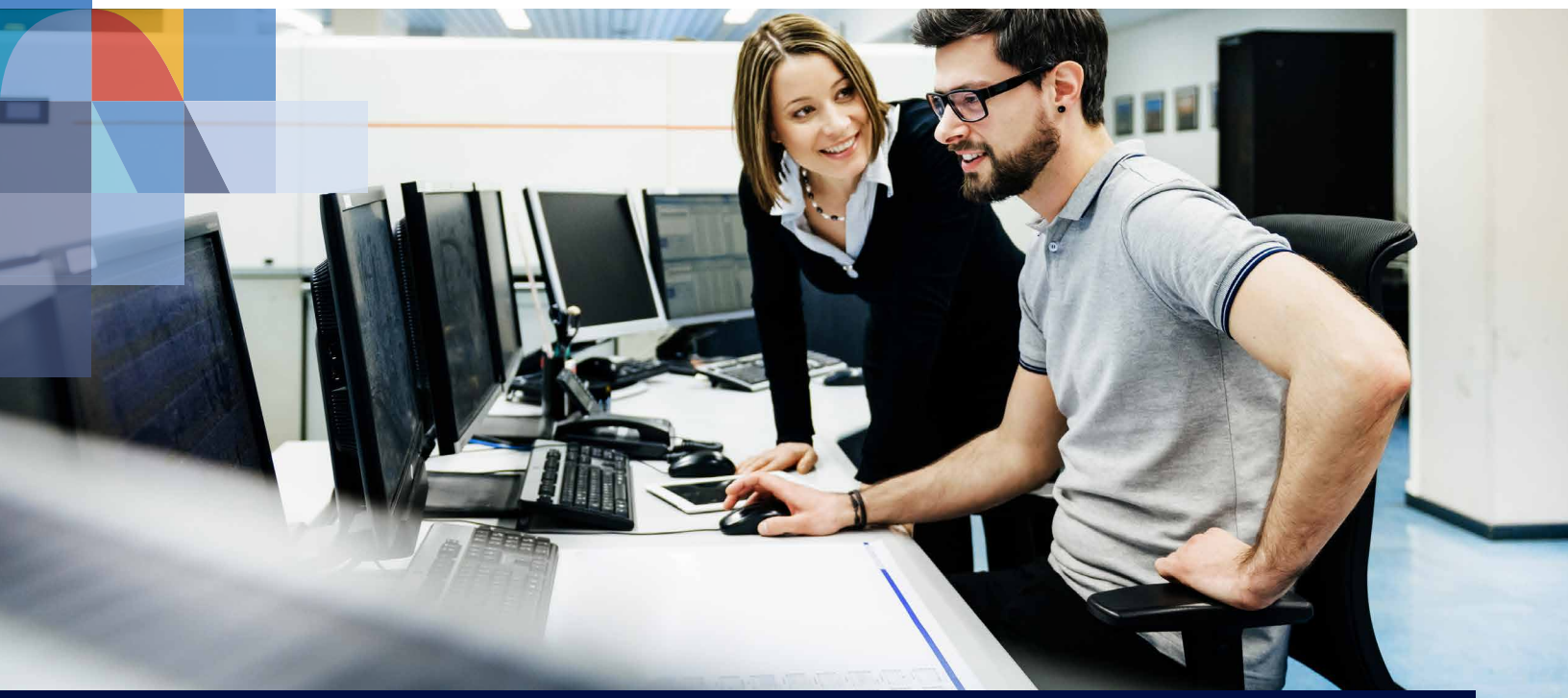
NTT DATA Services



WHITE PAPER | APPLICATION SERVICES

Continuous User Experience Engineering

JANUARY 2021

NTT DATA

Software methodologies

Software development methodologies play a vital part in any software lifecycle. These frameworks structure, plan and control the process of developing an information system. Though the first framework emerged in the late 1960s, it was only in the 1990s that the actual software development lifecycle (SDLC) evolved as a standard methodology. As such, SDLC can be considered the oldest formalized methodology framework for building information systems. The main idea of the SDLC is "to pursue the development of information systems in a very deliberate, structured and methodical way, requiring each stage of the life cycle from inception of the idea to delivery of the final system, to be carried out rigidly and sequentially," according to Geoffrey Elliott in his 2004 book, "Global Business Information Technology."*

NTT DATA ideated the **Continuous User Experience Engineering℠**, or CUE², methodology to push the user community as the key to an effective software outcome.

Before we delve into the details of CUE², let's discuss some of the standard software development methodologies currently in use, specifically:

- Waterfall model
- Large iterations
- Lean development methodology
- Agile software development
- Scrum development methodology
- Development and operations (DevOps)

Unlike traditional methods, such as waterfall, large iterations or agile, DevOps does consider user aspects. However, the level of user involvement is very low compared to what NTT DATA CUE² recommends.

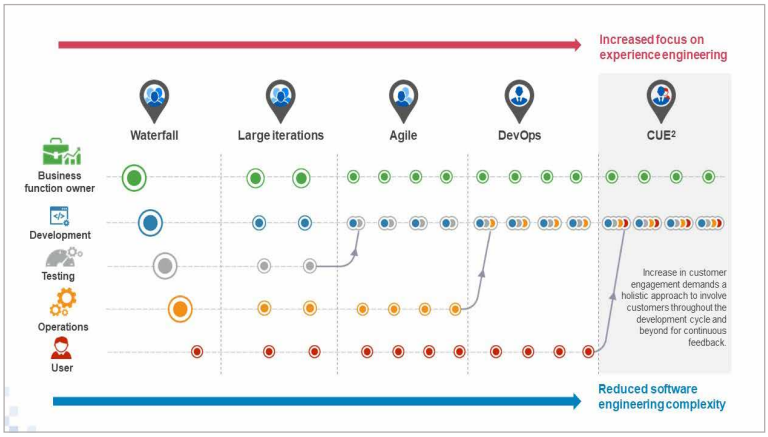


Figure 1: Various development methodologies and where CUE2 bridges the gap

Here are some of the pros and cons of these methodologies:

The most commonly used model, **waterfall** clarifies the software development process in a sequential flow, which means that any phase in the development process begins only if the previous phase is complete. Also, this development approach does not define a process for returning to the previous phase to handle requirement changes. The model is easy to understand, but it can only be used when very precise requirements are available, and it may not be applicable for maintenance projects. Waterfall's main drawback is that once an application is in the testing stage, it's impossible to make further changes. **Large iterations**, such as spiral, combine sequential flows and a prototype model. This framework is best used for projects that involve continuous enhancements and for larger projects that develop and deliver smaller prototypes and then enhance the model to make the larger software. Implementing this model requires experienced resources, due in part to the integral role of risk analysis. As a result, large iterations can be costly. User involvement is very low in this methodology.

The **lean development methodology** focuses on the creation of easily changeable software. It's more strategically focused than any other agile methodology. The goal of lean is to develop software in one-third of the time, with very limited budget and much less required workflow. One of this model's major disadvantages is that successful software development depends on the discipline of team members. Also, including a business analyst on the team is vital to ensure that the business requirements documentation is understood properly.

Agile software development is an approach used to design a disciplined software management process that also permits frequent alteration during project development. This methodology is adaptive and able to respond quickly to changing client requirements. Agile focuses on working software rather than documentation, so this approach may result in a lack of documentation.

The **scrum development methodology** can be applied to nearly any project, but it's best suited for development projects that are rapidly changing. Use this methodology for the speedy development of software that includes a series of iterations, with decision-making entirely in the hands of team members.

* "Global Business Information Technology: Systems Theory and Practice," Geoffrey Elliott, Addison-Wesley, 2004.

The main challenge with scrum is that the estimated project costs and time requirements may not be accurate. Also, it may not be appropriate for large projects.

A combination of development and operations, DevOps integrates all software functions from development to operations within the same cycle. Although switching to a DevOps methodology may bring a number of benefits, including accelerated time to market, faster user feedback due to more frequent releases, and improved productivity and customer satisfaction, it comes with certain challenges. For example, as development and deployment constraints are removed and programmers and system administrators gain more independence, the appropriate mechanisms for a feedback loop from operations to development must be set in place.

Over the years, software engineering complexity has been drastically reduced through state-of-the-art integrated development environments, rapid development utilities, collaboration software, code quality assessment toolkits and sophisticated unit testing frameworks, all of which are easily available with cost-friendly infrastructure. At the same time, there is tremendous demand for an enhanced, empathy-based experience in all walks of life, including the user interface in business-related software. This demand led to the birth of the CUE^2 SDLC. Figure 1 describes the various methodologies, from waterfall to DevOps, how CUE^2 bridges the gap from one methodology to another, and how an increased focus on experience engineering reduces the complexity of software engineering.

What is Continuous User Experience Engineering (CUE^2)?

The increase in customer engagement demands a holistic approach that involves users and key stakeholders throughout the development cycle and beyond for continuous feedback. CUE^2 addresses this need using three foundational phases.

All three phases are governed by the organization's policies and procedures. The inspire phase refers to strategy-related activities such as transformation and automation. Ideate brings teams together to execute those strategies. In this phase, teams break down the larger goals of the strategy into smaller, easily achievable objectives such as requirement specifications, user stories, use cases and features backlog. The outcome of the ideate phase is the implement phase, which focuses on coding, testing, deployment and monitoring. Figure 2 shows the full CUE^2 development lifecycle, built on the three phases:

Inspire considers automation, digital, business processes, software and experience engineering. The expectation is that every strategy and backlog item incorporates aspects related to these forces. Inspire considers the feedback from the current ecosystem of applications. CUE^2 proposes that feedback from application users must be the key to this process, and given as much (if not more) consideration as the feedback received from the engineering teams (in the form of a DevOps tool stack).

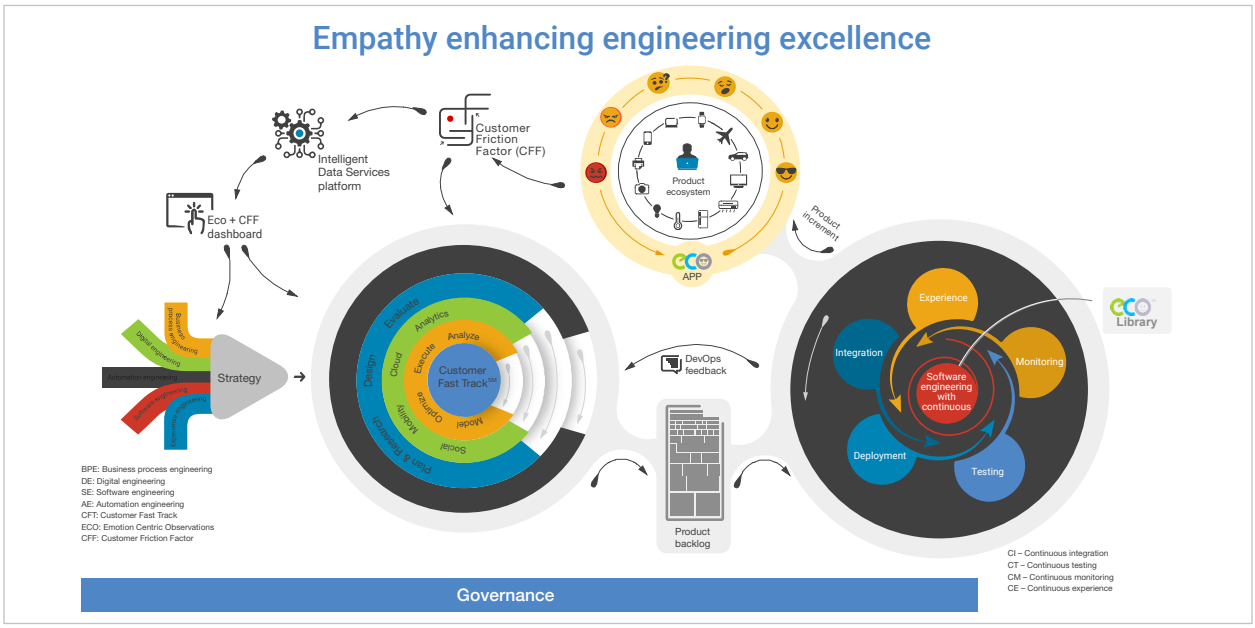


Figure 2: CUE^2 development lifecycle

Ideate considers NTT DATA Customer Fast Track, an innovative, high-speed requirements elicitation framework for effectively documenting the requirements that the downstream phases of the SDLC can use. This framework analyzes the forces of automation, experience, digital and business processes at a much deeper level. The expectation from this phase is to generate a prioritized, well-groomed backlog.

Implement considers a fully functional DevOps toolchain that enables continuous integration, continuous deployment, continuous testing, continuous monitoring and continuous experience. Today, traditional toolchains available in the market enable everything but continuous experience.

Emotion Centric Observations (ECO)

For continuous experience, NTT DATA has built utilities like Emotion Centric Observations (ECO). An application developed primarily to capture feedback from an end user and send it back to the development team, ECO is now available for existing applications. NTT DATA also plans to release ECO libraries to enable deeper user analysis of new and developing applications. Development teams will be able to embed ECO in business code to gather user feedback in a more effective way compared to former methods.

To enable feedback, the CUE[2] process ensures that ECO runs on all user desktops (Windows, Macintosh and Linux). We also recommend an NTT DATA Customer Friction Factor[SM] (CFF[SM]) assessment for all business-critical applications during this process. Feedback from ECO and CFF is fed into the NTT DATA IDS platform, which processes data to extract knowledge such as criticality and usage. Its intuitive graphical user interface handles analytical and advanced machine learning requirements. The platform also has an interface to load and profile data. Intelligent Data Services output is extended for strategy exercises or to initiate change management.

Figure 3 shows the components in a typical ECO environment:

- **User Journey Daemon.** A program that captures all user interactions in the application context.
- **Micro-Sentiment Engine.** A mechanism that enables the user to select a micro-sentiment (emoticons) and provide comments in the form of text, screenshots and/or audio.
- **Registry.** A server-side component that recognizes the applications or URLs registered for ECO.
- **ECO Launchpad.** A thin or thick client that runs on each user device and provides a facade for all the applications or URLs the user is entitled to use for enterprise business. Launch the ECO application to capture the feedback for an enterprise application.

continued on next page

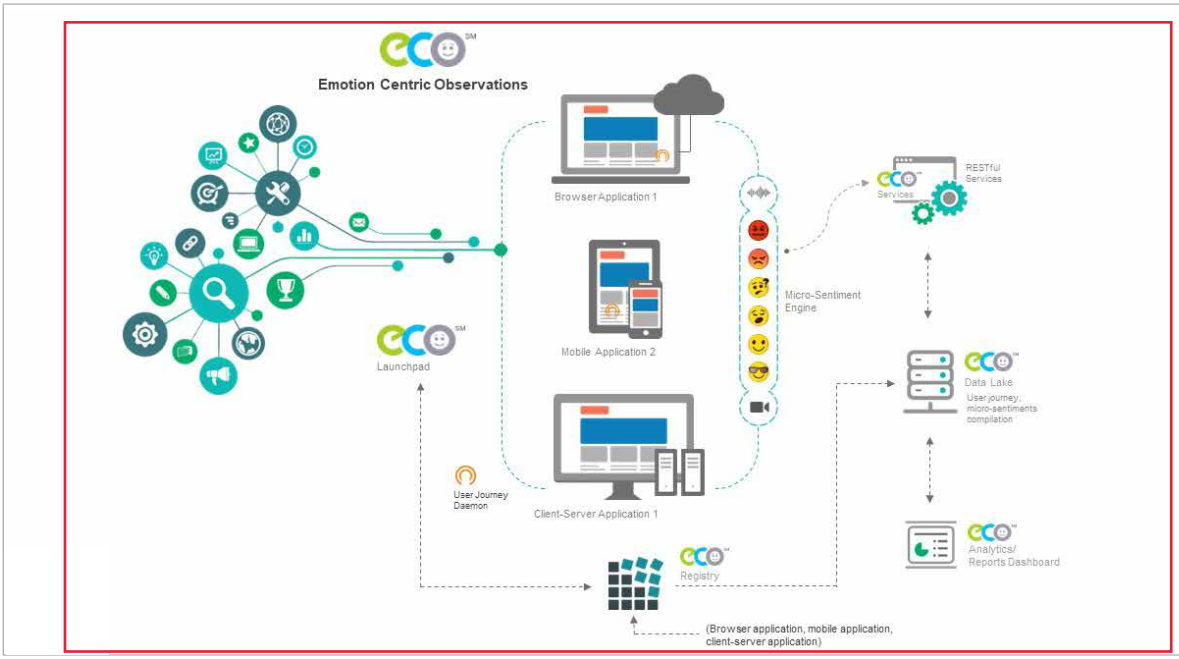


Figure 3: Components in an ECO environment

- **RESTful Services.** A set of RESTful services that supports extraction of registered applications and persistence of user journey and micro-sentiment data.
- **Data Lake.** A repository that compiles application-specific and user-related journey and micro-sentiment data.
- **Analytics/ Reports Dashboard.** This interface provides necessary feeds for change management of the applications.
- **ECO Library.** This collection is mainly used in applications for deeper user analysis.

Conclusion

All the existing software methodologies work well in their respective projects, depending on the nature of the project. Very often, the methodology best suited for a particular project may not be appropriate for another development. None of these methodologies is foolproof; each has its own pros and cons.

The consumerization of IT has increased the focus on experience engineering for enterprise applications. With demand for a better user experience growing, organizations need to ensure that experience is the key aspect in SDLC. That's where NTT DATA's CUE^2 shines. It combines the best practices from the agile and DevOps methodologies with an enhanced focus on the user. CUE^2 does not move away from agile/DevOps. Rather, it augments the DevOps toolset and agile best practices, aligns fully with the agile manifesto and introduces a fifth principle, Empathy Enhancing Engineering Excellence, to bring increased understanding to the user interface. CUE^2 focuses on the best software engineering methods, but for a truly phenomenal user experience all methodologies must be aligned — with the user in mind.

Visit **nttdataservices.com** to learn more.

NTT DATA Services, a global digital business and IT services leader, is the largest business unit outside Japan of NTT DATA Corporation and part of NTT Group. With our consultative approach, we leverage deep industry expertise and leading-edge technologies powered by AI, automation and cloud to create practical and scalable solutions that contribute to society and help clients worldwide accelerate their digital journeys.