Industrial Deployment of the TTCN-3 Testing Technology

Software developers face severe testing challenges today in terms of increasing system complexity, reduced time to market, and stringent quality requirements. These challenges are particularly prevalent in mobile telecommunications system development. TTCN-3 is a standardized testing language that offers a possible solution to many current testing challenges. However, introducing a new technology in a large organization is a complex process in itself. The authors describe their experience addressing these complexities with TTCN-3 at Nokia. This article is part of a special issue on Software Testing. Deiss, T.; Nyberg, A. J.; Schulz, S.; Teittinen, R.; Willcock, C. Software, IEEE 48-54 July-Aug. 2006.
Industrial Deployment of the TTCN-3 Testing Technology

Thomas Deiß, Andreas J. Nyberg, Stephan Schulz, and Colin Willcock,
Nokia Research Center

Risto Teittinen, Nokia Networks

The development of mobile telecommunications systems poses numerous testing challenges that have become more severe over time. We can categorize these challenges as increasing complexity, shorter time to market, and more exacting quality requirements. The seamless interoperability needed between different vendors’ equipment requires stringent testing. Ongoing product revisions, including new features in consecutive product releases, is another challenge. As an example of increased complexity, the number of specification documents has almost doubled from a second-generation (2G) system like Global System for Mobile Communication (GSM) to a third-generation (3G) system like Universal Mobile Telephony System (UMTS).

At the same time, the pressure to increase product quality is intense. For example, certain network elements in mobile communications systems should experience less than one second of downtime per year. These testing challenges translate into conflicting requirements that are difficult to reconcile and equally difficult to measure. At Nokia, we expected that the Testing and Test Control Notation 3 (TTCN-3) technology to address such product testing challenges better than other testing languages we’ve used in the past. However, introducing any new technology in a company as large as Nokia requires a significant amount of work.

We adopted a three-phase approach to introduce the TTCN-3 technology and resolve several key issues in the process—some technical and some organizational.

**TTCN-3 in brief**

A team of experts from major telecommunication companies, test tool vendors, and other organizations developed TTCN-3 and standardized it through the European Telecommunications Standardization Institute (ETSI).\(^1\)\(^-\)\(^4\)

TTCN-3 is similar to a programming language with specific features that support automated testing, such as test case *verdicts*, which express a test case’s outcome. TTCN-3 has specific rules for computing a test case’s overall verdict. TTCN-3 also provides various ways to express how to match data received from a *system under test* (SUT) and the parts of SUT data that are relevant to that match.
The main goals in TTCN-3’s development have been to define a notation that is:

- **well defined.** A formal semantics gives a single, precise meaning for any piece of TTCN-3 code on any machine.
- **easy to learn.** TTCN-3 looks like an imperative programming language. It’s also possible to produce TTCN-3 from graphical formats, such as message sequence charts.
- **flexible to use.** TTCN-3 supports specification of concurrent test-case behaviors, which lets developers create test components dynamically. It supports regular expressions, which are useful in testing text-based protocols. Also, TTCN-3 test systems’ implementation and integration aren’t tied to one specific programming language or operating system.
- **relevant to a range of application areas.** TTCN-3 supports message- and procedure-based communication. It applies to various testing phases, including software module, functional, and integration testing.
- **extensible with definitions from other languages.** TTCN-3 provides means to import definitions directly from other languages including IDL, XML, and ASN.1.

TTCN-3 has already proven its usefulness beyond the telecommunications industry. For example, the TT-Medal project includes industrial case studies in the automotive, railway, and financial domains (see the Web resources sidebar). One reason for this portability is TTCN-3’s independence from a specific methodology such as conformance testing.5

TTCN-3 abstracts from implementation details such as message-transfer syntax. The general test-system architecture in figure 1 shows a sample TTCN-3 code executable enclosed within other entities that provide these implementation details. The interfaces between the TTCN-3 executable and the surrounding entities are also standardized.1

The TTCN-3 runtime interface (TRI) hides the implementation details of the actual communication between the test system and the SUT. The platform adapter (PA) abstracts from the specific platform that executes the test system. The TTCN-3 control interface (TCI) lets users control and log test-case execution. In addition, the TCI abstracts from specific runtime system implementations and thus supports tool-independent development of encoders and decoders. These encoders and decoders help translate between the abstract value notation TTCN-3 uses and the transfer syntax needed to communicate with the SUT. (Stephan Schulz and Theofanis Vassiliou-Gioles describe instances of this test-system architecture.6)

**Competence development and transfer**

A critical part of successfully deploying a new technology in a large company is a well-coordinated plan for developing internal com-

---

**TTCN-3 Web Resources**

The following sites provide standard and implementation resources:

- **European Telecommunications Standardization Institute, e-Standardization portal:** http://portal.etsi.org
- **TT-Medal Project:** www.tt-medal.org
- **ETSI-eEurope Standardization Project for IPv6 Test Specifications Development:** www.ipt.etsi.org
- **TTCN-3 community homepage:** www.ttcn-3.org

The following companies offer tools for implementing TTCN-3 test systems:

- **Danet:** www.danet.com
- **OpenTTCN/Tester:** www.openttcn.com
- **Telelogic/Tau Tester:** www.telelogic.com
- **TestingTech/TTworkbench:** www.testingtech.com

---

**Figure 1. Testing and Test Control Notation 3 (TTCN-3) test-system overview.**
petence with the new technology and transferring it throughout the company. At Nokia, we chose a three-phase approach, with each phase lasting at least one year. In the first phase, we developed the tools and produced pilot systems, creating competence in a core team. In the second phase, we got the first product test systems working, thereby transferring competence to expert users. In the third phase, we transferred the technology more broadly to both Nokia product units and subcontractors. This approach proved very effective.

**Phase 1**

When Nokia began looking into TTCN-3, the standardization still hadn’t been finalized, so no tools or practical experiences with the language were available. It was possible to write TTCN-3 code, but we couldn’t utilize it because the tools that provide the runtime environments and compilers weren’t available.

In the first phase, a small team of researchers with various skills—for example, in compiler construction and test-system development—worked exclusively with the new testing technology. Results of this phase included numerous contributions to language standardization, a powerful TTCN-3 tool complying with Nokia’s needs, conceptual development of Nokia-wide test platforms, implementation of pilot systems for testing communication protocols, a style guide for test implementation, and training materials for power users—that is, test experts in particular product features. Engaging representatives from product testing in these activities was a critical factor for success in this first phase.

To ensure early availability of an industrial-strength TTCN-3 integrated development environment (IDE) containing a TTCN-3 analyzer, C compiler, and runtime environment within a GUI, we teamed up with a major tool developer. The partnership has been unusually close; developers from Nokia even participated in the tool development. This joint development let us provide early feedback on the IDE and react to tool problems rapidly. Depending on a specific test system’s needs, we’ve been able to fix critical bugs in one day. The tight feedback and error-correction loop was important early in the technology deployment. It let us develop the tool according to feedback from real usage but didn’t allow problems in the tool development to interfere with real usage.

The standards themselves were the only materials available when we started with TTCN-3. But their intention was to define a language, not teach it. We therefore had test engineers attend a two- to three-day hands-on testing course to ensure quick acceptance of the language. We designed the course to gradually introduce the language’s most relevant aspects. It also covered the basics of test-system operation and included practical exercises. We chose the Session Initiation Protocol as the basis for the course exercises. SIP is a call-control protocol for voice over IP. Its textual encoding and simple semantics let us simplify concepts so that course participants could focus on the actual testing concepts instead of protocol oddities. Despite our simplifications, the test engineers perceived SIP as a real-life example.

Finally, we introduced test engineers to their specific test system with another one-day course.

**Phase 2**

Nokia’s early internal development of methodologies and the extra effort spent creating effective, practical training materials proved beneficial in the second phase, which focused on technology transfer. In this phase, the testing experts made their knowledge available throughout the company by hosting TTCN-3 courses and seminars on demand, writing the first textbook on the topic, and providing a discussion forum for product testing. In addition, the TTCN-3 experts began investigating applications of this technology in nontraditional areas such as software module testing and CORBA-based systems testing. It also proved important in this phase to continue sharing information beyond company boundaries—for example, in standardization activities, ETSI testing task forces, the European TT-Medal project, and TTCN-3 user conferences.

**Phase 3**

The third phase concentrated on transferring TTCN-3 competencies to external partners, especially in the area of test-system development. Our objective was to prepare Nokia for a large-scale migration. Through the transfer of tasks, such as maintaining existing test platforms and developing new ones, the original team of test experts was able to switch its role to consulting with the Nokia product test teams.
At this point, the question “Why should we use TTCN-3?” changed to “When can we start using TTCN-3?”

**Test-platform development**

At the time we deployed TTCN-3, Nokia R&D test managers were interested in harmonizing the company’s test systems. Test-system development was scattered across the organization, different groups used different test script languages, and cooperation among developers of individual test systems and toward long-term development was modest. To improve this situation, we based the first TTCN-3 test systems on the concept of test platforms. A test platform speeds test-system development and provides homogenous systems for the end users.

We adopted the general TTCN-3 test-system architecture in figure 1 as a framework for developing two test platforms: one focused on a generic way to use typical mobile communications protocols in the SUT adapter (SA). The second emphasized the codecs for the text-based protocols such as SIP and using protocols such as UDP and TCP for communications with the SUT. From previous experience on other test-system developments, we identified components for reuse between test systems within a test platform and even across test platforms. The key reuse components are as follows:

- **Specific protocol codecs and libraries**, for developing codecs for whole protocol classes. Many protocols within 2G and 3G systems use standardized encodings, so the TTCN-3 tool could automatically generate the corresponding codecs.
- **SAs connecting the test system with the SUT**, for developing variants for different application areas.
- **Platform adapters**, for implementing timers and integration of external functionality such as ciphering functions.
- **Test-logging mechanisms**, for improving the readability of test-case and test-suite execution logs.
- **Test-control mechanisms**, for executing test cases and managing test suites.

These libraries provided commonly used type definitions and functionality, such as a test-component synchronization mechanism. We also found that the test systems could reuse even parts of test cases. Developing test platforms helped us deploy TTCN-3. After we’d built a first test system as an instance of a test platform, we could rather easily build further test systems by combining the reusable components. The general test-system architecture (see figure 1) facilitates the development of test platforms.

**TTCN-3 deployment**

The last five years have revealed the challenges of deploying a new test specification language in a large organization. It’s not just a matter of introducing the language; it also involves a range of other technical, organizational, and human factors.

**Current status**

For Nokia, TTCN-3 provides a feature set that’s more efficient for test-case development than its predecessors. The core TTCN-3 tools, including compilers and editors, are now mature and enable more productive test-case development and execution. However, advanced features, such as test management, still need development to further improve efficiency and usability.

Nokia’s TTCN-3 deployment has seen steady growth in users. Figure 2 represents the number of TTCN-3 users between 2001 and the first quarter of 2006. In 2001, the IDE was under development and only a few engineers used TTCN-3. This changed signif-

![Figure 2. Estimated number of TTCN-3 users at Nokia between first quarter 2001 and first quarter 2006.](http://www.libsou.com)
significantly beginning in 2003, when we achieved reasonable confidence and deployed the first product-test systems. These systems acted as showcases for other test teams. During 2004, we deployed a few large test systems, and growth continued rapidly in 2005. We expect significant growth in the number of TTCN-3 product-test systems and users over the next 12 to 18 months.

So far, most users have worked in functional or conformance testing of systems or subsystems.

TTCN-3 technology doesn’t need significant promotion within Nokia today. In fact, TTCN-3 has become a favorite testing language.

External test suites

Some externally developed test suites are also relevant for Nokia. These suites, developed mostly by standardization bodies to certify products, are already available or under development for telecommunications, the Internet, and automotive equipment. For example, ETSI has developed an extensive TTCN-3 IPv6 test suite (www.ipt.etsi.org) for the eEurope consortium. The first release of this executable test suite was published earlier this year; it contains more than 200 test cases for certifying core IPv6 functionality, as defined by the IPv6 logo committee.

The existence of such external test suites highlights the unifying character of TTCN-3 as a test language. Because TTCN-3 lets us unambiguously execute tests independent of a particular testing tool, it’s well suited to define conformance and interoperability test suites for equipment from different manufacturers.

Coping with legacy test systems

Nokia has used some of its legacy test technologies for over a decade. We can’t instantly replace the competence related to them with TTCN-3 competence, so training has played a significant role in this stage. With the help of an external company, we developed a translator for converting existing test scripts to TTCN-3.

However, reusing these modules requires extra effort. It’s difficult to predict whether the required effort is insignificant, moderate, or major. Some of our test suites are large and complex, so we’ve put significant effort into making the migrations smoother.

Results and lessons learned

The experiences of people involved in the TTCN-3 technology development and deployment taught us many lessons. To complement this, we conducted a survey among the end users within TTCN-3 test teams. Table 1 shows the survey’s key results. The numbers are based on input from approximately 70 percent of the active TTCN-3 testers in the beginning of 2006 (see figure 2). We included these results in elaborating the most important lessons learned.

Unified testing technology

Because TTCN-3 is a unified testing technology, test team managers understood that it

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Key user survey results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Question</strong></td>
<td><strong>Strongly agree (%)</strong></td>
</tr>
<tr>
<td>TTCN-3 makes it easier for me to do what I want compared to the test implementation or the scripting languages I have previously used.</td>
<td>45.5</td>
</tr>
<tr>
<td>I am more comfortable with the TTCN-3 language than with other test implementation or scripting languages.</td>
<td>27.3</td>
</tr>
<tr>
<td>In my opinion, Nokia-wide coordination has reduced my test system development costs.</td>
<td>28.6</td>
</tr>
<tr>
<td>TTCN-3 training has been critical to using the language quickly.</td>
<td>20.0</td>
</tr>
<tr>
<td>TTCN-3 training has served well in my everyday work.</td>
<td>0.0</td>
</tr>
<tr>
<td>TTCN-3 has enabled me to write more complicated test cases than the test implementation or the scripting languages I have used before.</td>
<td>45.5</td>
</tr>
<tr>
<td>Reuse of my team-specific test implementation artifacts has increased since adopting TTCN-3.</td>
<td>22.2</td>
</tr>
<tr>
<td>Reuse of other project’s test implementation artifacts has increased since adopting TTCN-3.</td>
<td>40.0</td>
</tr>
</tbody>
</table>
enables the development of harmonized test systems across Nokia. As a consequence, TTCN-3 doesn’t need any more significant promotion within the company. Also, managers now recognize that TTCN-3 is a basic test engineering skill. In large organizations, this means engineers can move between test organizations without having to learn a new testing language.

A unified testing technology might bring further advantages such as reusing test code between different test phases. However, as our survey results indicate, TTCN-3 code reuse between test teams isn’t yet significant. We attribute this partly to a lack of awareness of TTCN-3 code availability. Reuse needs ongoing support, so we’ve set up an internal open source project to make reusable code more visible to the test engineers.

Test platform efficiencies

The test-platform concept has proven effective. In practice, the platforms have sped up the development of new TTCN-3 test systems as well as the extension of existing systems. We can set up new test systems in a matter of days by combining building blocks with the needed functionality. The test-platform approach has also provided a consistent test-system appearance for the end users. Nevertheless, we should have better coordinated and aligned the two test-platform developments to make it even more effective and to avoid merging and reimplementing some features. We also need to focus more on documentation to make the test platforms easier to use.

Tool chain integration

Test specification has become more effective using TTCN-3’s well-integrated tool chain. In some cases, turnaround time for bug correction has decreased by 40 percent. This is a significant improvement for large test suites, which can contain hundreds or even thousands of test cases. The improvements have been larger for more complex functionality. (Other companies have reported similar improvements.)

Although the tool chain has been well integrated, our test engineers have indicated that its usability needs improvement. We’ve learned it’s important to pay attention to usability earlier on. In addition, some essential tools, such as TTCN-3 debuggers, are still missing from our tool set. In general, tool development for a new language is complicated. Early contact with end users and our use of prototype tools to develop pilot test systems have helped us reach the tool maturity needed for product-test systems. There were also many efficiencies in some of us simultaneously developing the tools and applying them in the pilot test cases. However, we gave some of the tools or tool extensions to end users too early, and regaining their lost confidence has been difficult.

Positive user response

Nevertheless, our end users have a largely positive view of TTCN-3. Eighty percent of the survey respondents reported that it’s easier to implement complex test cases with TTCN-3 than with other testing or script languages they’ve used before; 50 percent of the respondents rated this ease as one of the most important benefits of the TTCN-3 technology. In addition, 90 percent indicated that they prefer TTCN-3 over other testing languages and scripts.

A key to this success has been our internal TTCN-3 training programs, where we teach TTCN-3 by applying it to a protocol that the test engineers must test. Participants have reported that they’ve learned TTCN-3 technology within a month to a level that feels comfortable for their everyday work.

Reports from the test teams deploying TTCN-3 indicate the most important reason for choosing it is the ability to implement more complex test cases than they could with the test languages they’d previously used. Other reasons included increased reuse, a better tool set, and TTCN-3’s standardization, which makes it independent from tool vendors and platforms.

The testing challenges we’ve described aren’t unique to the telecommunications industry. They translate into testing technology requirements that apply to all software-driven enterprises. Because testing typically accounts for 25 to 50 percent of software development costs, improved methodologies have the potential to pay off big. Nokia’s experience in the telecommunications domain shows that TTCN-3 offers significant efficiencies for industrial product testing.
through its standardized language and interfaces. It's proven strong enough to help build large and complex test systems containing thousands of test cases; several systems are in use at different Nokia business units for product testing today. Some R&D test organizations have started using TTCN-3 for other kinds of testing than functional testing, but whether it will yield significant benefits compared to other notations in those areas remains to be seen.

Acknowledgments

Our work wouldn't have been possible without the contributions of numerous people, of whom we can name just a few. We thank our colleagues Federico Engler, Sami Heinonen, Olli Hentinnen, Stefan Keil, and Stephan Tobies; users who provided constructive criticism (Dan Cimpona and Gregory Ek); and collaborators from external institutes and companies (Ina Schieferdecker, Theofanis Vassiliou-Gioles, and Anthony Wiles). We also thank the anonymous reviewers for their constructive comments on improving this article.

References