In 2015 the senior-year course "Real-Time and Concurrent Systems at a High Level of Abstraction" was introduced at the Australian National University (ANU). This course expands students' knowledge into understanding and managing the principles behind concurrent programming. For example, students are able to design and implement real-time software at an abstraction level that reflected their needs by writing Ada code and simulating it in a high-level environment.

**Spotlighting a GAP Member:**

Our current GAP Member is Uwe Zimmer, an expert in both Ada and photorealistic 3D rendering. His research and development focus on the integration of Ada and photorealistic rendering to create immersive virtual environments.

**BNP Paribas Report:**

In a recent report from the BNP Paribas report, it was highlighted that Ada is a strong choice for financial services due to its robust standards and proven reliability.

**Tech Days 2017 Australia:**

Tech Days 2017 was held at the Australian National University, Canberra. The event featured various demonstrations, presentations, and networking opportunities for attendees interested in the latest advancements in Ada and related technologies.

**AdaCore Open Source Tools:**

AdaCore offers various open-source tools that are available for download. These tools include AdaIDE, a graphical Ada development environment, and GNAT Pro, a comprehensive Ada development environment.
The 3rd place prize of €1000 was awarded to Shawn Nock for his Blue - tooth Beacons “iBeacons” project. The target was a Nordic Semiconductor Radio (Bluetooth Low Energy) with a custom development board.

The winners of the first annual AdaCore Prize for Invention (Cortex M7). The software analyzes and monitors Ethernet traffic by reading EtherScope project running on an STMicroelectronics STM32F746 board. The software uses Ada, and the entries were evaluated against how Ada and SPARK can help produce reliable embedded software, and the submitted projects ranged from food technology to archeology. Open to individual contributors and small teams, the prize was targeted to an ARM Cortex M or R processor.

Lead of Business Development and Technical Account Management

Quentin, tell us about the AdaCore software solutions that are used in your role? You have been involved in many projects and there is a wide variety of applications. Could you share some examples of your work?

The AdaCore technology is now being used in more than 60 companies, including some of the largest companies in the world. We provide a wide range of solutions to support our customers in their development process. For example, we work with automotive companies to help them develop safety-critical systems, and we also work with organizations in the aerospace and defense industries to help them develop robust and reliable systems.

What kinds of challenges and opportunities do you foresee as you move forward?

The challenges we face are driven by the rapidly evolving requirements of our customers. We need to stay ahead of the curve and develop new solutions that meet the changing needs of the industry. One of the biggest challenges is the ability to integrate our solutions with existing systems in a seamless manner. Another challenge is the need to ensure that our solutions are secure and compliant with industry standards.

The opportunities are related to the growing demand for our solutions. The demand is particularly strong in areas such as automotive, aerospace, and defense. We are seeing a increase in the number of projects that require our solutions, and we are working hard to meet that demand.

What are your thoughts on the future of Ada?

Ada is a highly reliable programming language that is well-suited for developing critical systems. In the future, we believe that Ada will continue to be an important language, particularly in the areas of safety-critical systems.

How do you see AdaCore evolving in the future?

We believe that AdaCore will continue to evolve as the programming language and tools support continue to improve. We are committed to providing our customers with the most advanced solutions to help them succeed in their development projects.
Based on GCC 6 and GDB 7.10

**GPRbuild**
- Improved code generation and support for additional blocks
- Process of the Loop: new perspective
- Contextual hints

**GNATbench**
- "No False Positives" mode, to suppress likely false alarms
- Support for type invariants

**GPS**
- XMLAda, GNATcoll and AWS come precompiled with native GNAT
- Improved elaboration order algorithm
- Model-Level debugging
- Improved proof automation

**GNATtest**
- The winners of the first annual Make with Ada contest were announced.
- The contest, which was sponsored by AdaCore and held in Boston and Paris, was open to individual contributors and small teams.
- Participants had to use Ada or SPARK to develop an embedded software application that demonstrated their creativity and technical expertise.
- The contest featured three prize categories: the Gold Medal for the best overall project, the Blue Shield for the most innovative project, and the Red Shield for the most technically challenging project.
- The Gold Medal was awarded to a team from the University of California, Berkeley, for their project "Blue Shield: A Miniature Race Car for the Internet of Things".
- The Blue Shield was awarded to a team from Stanford University, for their project "Ravenscar: A Safe and Efficient Framework for Real-Time Systems".
- The Red Shield was awarded to a team from the University of California, San Diego, for their project "Turing: A New Approach to Dynamic Analysis".

**OpenOCD**
- Redesigned handling of bookmarks
- Improved debugger support: new default perspective, variables view
- Improved GNATcoverage integration
- Improved support for stubbing
- Export from library can be limited to symbols in its interface

**QGen**
- Enhanced support for internal
- Generation of reader and writer for each block in the application
- Process of the Loop: new perspective
- Contextual hints

**SPARK**
- New mode statistics for proof results
- Improved counterexamples (arrays, quantifiers, enumerations)
- Finer granularity in proof of conjunctions
- Generation of loop invariants for frame conditions
- CodePeer engine available as an additional prover
- Improved analysis of unchecked conversions, access checks, test
- Loop unrolling and better handling of static array bounds

**QGen**
- Enhanced support for internal
- Generation of reader and writer for each block in the application
- Process of the Loop: new perspective
- Contextual hints

**SPARK**
- New mode statistics for proof results
- Improved counterexamples (arrays, quantifiers, enumerations)
- Finer granularity in proof of conjunctions
- Generation of loop invariants for frame conditions
- CodePeer engine available as an additional prover
- Improved analysis of unchecked conversions, access checks, test
- Loop unrolling and better handling of static array bounds

**QGen**
- Enhanced support for internal
- Generation of reader and writer for each block in the application
- Process of the Loop: new perspective
- Contextual hints

**SPARK**
- New mode statistics for proof results
- Improved counterexamples (arrays, quantifiers, enumerations)
- Finer granularity in proof of conjunctions
- Generation of loop invariants for frame conditions
- CodePeer engine available as an additional prover
- Improved analysis of unchecked conversions, access checks, test
- Loop unrolling and better handling of static array bounds

**QGen**
- Enhanced support for internal
- Generation of reader and writer for each block in the application
- Process of the Loop: new perspective
- Contextual hints

**SPARK**
- New mode statistics for proof results
- Improved counterexamples (arrays, quantifiers, enumerations)
- Finer granularity in proof of conjunctions
- Generation of loop invariants for frame conditions
- CodePeer engine available as an additional prover
- Improved analysis of unchecked conversions, access checks, test
- Loop unrolling and better handling of static array bounds

**QGen**
- Enhanced support for internal
- Generation of reader and writer for each block in the application
- Process of the Loop: new perspective
- Contextual hints

**SPARK**
- New mode statistics for proof results
- Improved counterexamples (arrays, quantifiers, enumerations)
- Finer granularity in proof of conjunctions
- Generation of loop invariants for frame conditions
- CodePeer engine available as an additional prover
- Improved analysis of unchecked conversions, access checks, test
- Loop unrolling and better handling of static array bounds

**QGen**
- Enhanced support for internal
- Generation of reader and writer for each block in the application
- Process of the Loop: new perspective
- Contextual hints

**SPARK**
- New mode statistics for proof results
- Improved counterexamples (arrays, quantifiers, enumerations)
- Finer granularity in proof of conjunctions
- Generation of loop invariants for frame conditions
- CodePeer engine available as an additional prover
- Improved analysis of unchecked conversions, access checks, test
- Loop unrolling and better handling of static array bounds

**QGen**
- Enhanced support for internal
- Generation of reader and writer for each block in the application
- Process of the Loop: new perspective
- Contextual hints

**SPARK**
- New mode statistics for proof results
- Improved counterexamples (arrays, quantifiers, enumerations)
- Finer granularity in proof of conjunctions
- Generation of loop invariants for frame conditions
- CodePeer engine available as an additional prover
- Improved analysis of unchecked conversions, access checks, test
- Loop unrolling and better handling of static array bounds

**QGen**
- Enhanced support for internal
- Generation of reader and writer for each block in the application
- Process of the Loop: new perspective
- Contextual hints
Version 17 Product Releases

New Product Numbering and Release Scheme
To simplify and unify the version numbering and annual release cycle for AdaCore products, a new numbering scheme was adopted in February 2017. Each product will have a version number of the form ‘17.x’ where ‘17’ identifies the major AdaCore product being released (1 for GNAT Pro, 2 for CodePeer, etc.) and ‘x’ the minor release cycle. Thus 17.1 is the version number for each AdaCore product being released in the first half of 2017, 17.2 is the second half of the year, 17.3 is the version number for each AdaCore product being released in the second half of 2017, and so on.

GNAT Pro
• GNAT Pro (version 17.1)
  • Improved elaboration order algorithm
  • Support for incidental and stub coverage confinement
  • Improved elaboration order algorithm
  • Support for incidental and stub coverage confinement
  • Improved elaboration order algorithm
  • Support for incidental and stub coverage confinement

QGen
• Enhanced web interface
• Improved generation and support for additional blocks
• Process for the Loop-Game (Quentin Ochem)
• Mobile-Level debugging
• Better integration with external code

CodePeer
• "No False Positives" mode: to suppress false false alarms
• Loop-simplifying and better handling of static array bounds
• Improved handling of static array bounds
• Improved handling of static array bounds
• Improved handling of static array bounds

GPS
• Global objects used in subprograms displayed in annotations
• Exception messages now displayed when stepping on an exception
• Support for fast context changes when stepping a code view
• Exception messages now displayed when stepping on an exception
• Support for fast context changes when stepping a code view

SPARK
• New mode statistics for proof results
• New mode statistics for proof results
• New mode statistics for proof results
• New mode statistics for proof results

GNATcoverage
• Exception messages now displayed when stepping on an exception
• Support for fast context changes when stepping a code view
• Exception messages now displayed when stepping on an exception
• Support for fast context changes when stepping a code view

New Selections
• Support for additional blocks
• Process for the Loop-Game (Quentin Ochem)
• Mobile-Level debugging
• Better integration with external code
• "No False Positives" mode: to suppress false false alarms
• Loop-simplifying and better handling of static array bounds
• Improved handling of static array bounds
• Improved handling of static array bounds
• Improved handling of static array bounds

Professional training
Public Ada Course in the UK, April 2017
This course is an advanced fundamentals of Ada held during April 24–25, 2017 in UK. The course, which is open to both Ada and other developers, mainly focuses on AdaCore GNAT Pro technology, covers the essentials of software development with Ada and the major AdaCore products: GNAT Pro, CodePeer, GNATmake and GNATcoverage. The training covers all the major new features announced in the GNAT 17 product release.

Even for those who are already familiar with Ada and GNAT technology, this course will help them refresh and update their knowledge in Ada

What’s included in the course
The course is 2 days long (14 hours in total) covering all major features of the Ada and GNAT technology.
Students use Ada to solve problems related to concurrent and distributed systems and how to solve them. Dr. Zimmer reports that the students have come to appreciate Ada's protected object type and how it is necessary for device drivers to write and for concurrent systems to ensure device drivers are properly synchronized.

Ada is also a principal language in a second-year course at the ANU, “Concurrent and Distributed Systems.” Ada has been used in this course since 2000, and the Ada tasking model has been instrumental in expanding the class size in 2017.

Owing to the success of this course, Dr. Zimmer has decided to continue teaching Ada in the course “Introduction to Embedded System”, conducted by Dr. Uwe Zimmer, in undergraduate computer science programs. The course at the ANU, “Concurrent and Distributed Systems,” is a government and industry open avionics standard for making military aircraft safer, more capable, and more reliable than constructs such as SWaT and MIL-STD-188. Ada has been used in this course since 2000, and the Ada tasking model has been instrumental in expanding the class size in 2017.

AdaCore is participating in the O’PAVES project (Open Platform for Autonomous Vehicle Systems), an effort funded by the European Union through the FP7 (Lagrange) consortium. The overall project goal is to develop a technology transfer to bring high reliability to the emerging domain of autonomous vehicle systems. In particular, to ensure safety and reliability, system designers need to ensure that the software applications they develop successfully in high-intensity application areas such as aerospace and rail. O’PAVES is using AdaCore’s tools to produce an open platform for developing and verifying software-intensive systems that will be used by students and researchers. The project will end in November 2017 and will complete in October 2018.

SIAGA HILT Workshop Summary

ACM SIGAda’s HILT (High Integrity Language Technology) workshop was held on October 6 and 7 in Pittsburgh, with a focus on Model-Based Development and Code-Based Program Assumption. Increasing the rigor of model-based development is a science across the breadth of the industry. To build formal contracts (such as those found in Ada2005) in the generated code is not only desirable; a technology demonstrator that can help developers verify that the software applications they develop successfully in the original system requirements. AdaCore was a Gold Sponsor of Ada2005, and further effort will be put into working on this project.

NIST Report on Software Vulnerabilities

A December 2016 report from the National Institute of Standards and Technology (NIST) addressed the types of software vulnerabilities and vulnerabilities that are prevalent in software systems. The report was based on a survey of over 100 organizations that use Ada. The report found that AdaCore’s tools are widely used in the industry and that other platforms can be used with similar success.

SPARK Webinar

On December 12, 2016, Kenichi Miyazaki (PMO Product Manager at AdaCore) and Rod Downham (Director, Product Line Consultant) conducted a webinar titled “Building High-Assurance Software Without Breaking the Bank.” The webinar covered the SPARK language and toolset technology and showed how it enables the development of high-assurance systems. The webinar was a summary of SPARK experience from a number of government and commercial projects, and explained an overview of the technology and how it is used in an engineering environment as a practical complement to its teaching. A recording of the webinar is available at https://www.adacore.com/products/spark/webinar.html.

Tech Days 2017 Australia

Tech Days 2017 Australia, Canberra, Australia, March 14–18, 2017 / Nuremberg, Germany, March 20–23, 2017

techdays.com.au/techdays/

AdaCore is an exhibitor at this event.

April 24–28, 2017 / Yeovil, UK

www.adacore.com/events/adacore-pmday/

TU-Automotive Detroit 2017

June 12–15, 2017 / Detroit, MI

www.tu-automotive.com

Euro Deleting 2017

November 29–30, 2017 / Basel, Switzerland

www.adacore.com/events/euroDeleting/
Ada is used in several courses at the Australian National University (Canberra, Australia), and recent experience has confirmed Ada’s benefits as a language to teach engineering techniques for the most demanding concurrent systems and how to solve them. Dr. Zimmer reports that the students have come to appreciate Ada’s protected object mechanism, which is simpler and more reliable than constructs such as monitors. The protected object mechanism eliminates the most common form of synchronization errors by checking for intent conflicts at run-time. The design of the protected object mechanism is based on real-time systems experience, and it is significantly simpler than alternative approaches.

A November 2016 report from the National Institute of Standards and Technology (NIST) is published twice a year. The report, titled “A Survey of NIST’s Work in Conformance Testing for Security and Privacy of Information Technologies,” was extended to cover hardware-level real-time programming. This includes Ada programming concepts at a higher level of abstraction while retaining the necessary low-level control. The GNAT technology now supports the gcc AddressSanitizer utility, which detects run-time memory errors such as out-of-bounds references and use-after-free bugs. Although many of these memory errors are prevented because of Ada’s static analysis, debuggers can still view live program status where the errors occur.

To use AdaLinkSanitizer, just add `-fsanitize=address` to the compiler’s command line. The sanitizer is completely transparent to the compiled application, except that it will calculate and report run-time errors. The results are not saved to memory, and no interruption occurs whenever a memory error is detected. The GNAT technology now supports the gcc AddressSanitizer utility, which detects run-time memory errors such as out-of-bounds references and use-after-free bugs. Although many of these memory errors are prevented because of Ada’s static analysis, debuggers can still view live program status where the errors occur.

The GNAT technology now supports the AdaLinkSanitizer tool, which emits a list of memory errors such as out-of-bounds references and use-after-free bugs. Although many of these memory errors are prevented because of Ada’s static analysis, debuggers can still view live program status where the errors occur.

**Buffer overflow**

The buffer overflow problem is widely considered to be one of the most common types of buffer overflows. A buffer overflow is a situation where data is read from a buffer that is not sufficiently sized to hold the entire buffer.