Notes

1 Introduction and Overview

1. In 1997 the SEI replaced the Software CMM with the Capability Maturity Model Integration (CMMI).
2. When using the (GPS) – integrated development environment or GNATbench (the Ada plug-in for Eclipse), the check and flow analyses are combined into the *examine* command.

2 The Basic SPARK Language

1. The variable `Letter`, declared as subtype `Uppercase` defined on page 45, has a domain of the 26 uppercase letters.
2. A different identifier with the same name.
3. Assignment and equality testing are not available for Ada's limited types. A limited type is a type that includes the reserved word **limited** in its definition or in the definition of a component of a composite type.
4. With SPARK, each instantiation is individually verified.

3 Programming in the Large

1. Ada's operators are `and`, `or`, `xor`, `=`, `/=`, `<`, `<=`, `>`, `>=`, `+`, `-` , `&`, `*`, `/`, `rem`, `mod`, `**`, `abs`, and `not`.
2. This sharing of private data is similar to the *friend class* notion found in some object-oriented programming languages.
3. Elaboration order is a significant problem in C++ and Java, programs where it is commonly called the *static initialization order fiasco* on programmer forums.

4 Dependency Contracts

1. Roughly, a *static expression* is an expression that can be evaluated by the compiler. See the Ada Reference Manual (2012), Section 4.9, for more details.

2. It does mean an ineffective value computed by the caller might appear effective if it participates as an input in a flow dependency that does not actually exist.
3. See the *SPARK 2014 Toolset User's Guide* (SPARK Team, 2014b).

5 Mathematical Background

1. Arguments in this context are usually called *conjectures* (potential theorems).
2. The operator ∈ is for set membership. The expression $x \in U$ is read "x is a member of the set of all humans alive today."

6 Proof

1. For an explanation of Ada's exceptions and exception handlers, see Dale and McCormick (2007) or Barnes (2014).
2. Examination of the subprogram may also fail if it is not sufficiently implemented to honor its data and flow dependency contracts.
3. The older SPARK 2005 does have some support for tasking.
4. The assume assertion is special. See Section 6.3.
5. The N-th Fibonacci number, $F(N)$, is given by $F(N-1)+F(N-2)$, where $F(0)=0$ and $F(1)=1$ are base cases.
6. In this example, the procedure only searches arrays of exactly 100 integers. In Section 6.7 we show a generic version of this procedure that is more general.
7. Recall that limited types cannot be copied.
8. *Inline expansion* is a process in which the compiler replaces subprogram calls with copies of the subprogram. This substitution typically improves the execution time and stack memory usage; however, it also increases the size of the program. The aspect Inline is used to request this expansion.
9. At the time of this writing SPARK does not support type invariants. However, support is planned for a future version of SPARK.
10. At the time of this writing SPARK does not support dynamic predicates. However, support is planned for a future version of SPARK.
11. For example, 192.168.56.2.
12. The algorithm can be supplemented to deal with single digit divisors (Knuth, 1998), but we do not do so here.
13. Secure hash algorithms are used with, for example, digital signature algorithms to provide strong data integrity checks.

7 Interfacing with SPARK

1. *Library level* means that the program unit is not nested within another program unit. It may be compiled separately and referenced in with and use clauses.
2. For an explanation of Ada's pointers, see Dale and McCormick (2007) or Barnes (2014).
3. For an explanation of Ada's exceptions and exception handlers, see Dale and McCormick (2007) or Barnes (2014).
4. For an explanation of Ada's automatic finalization, see Dale and McCormick (2007) or Barnes (2014).
5. For a full treatment of using Ada to interact with hardware, see McCormick, Singhoff, and Hugues (2011).

6. McCormick et al. (2011) provide a complete discussion on interacting with hardware in Ada.
7. Thanks to Angela Wallenburg, Altran UK, and Yannick Moy, AdaCore, for this example.
8. The complete syntax for abstract state aspects is given in Section 7.1.4 of the *Spark 2014 Reference Manual* (SPARK Team, 2014a).
9. All of the source code for this example is available on the http://www.cambridge.org/us/academic/subjects/computer-science/programming-languages-and-applied-logic/building-high-integrity-applications-spark?format=PB.

8 Software Engineering with SPARK

1. The postcondition was not proven when function `Empty` was written as an ordinary function. The SPARK tools do not generate postconditions for subprograms. The proof succeeded when a postcondition was added to the body of the ordinary function. Because the result of an expression function is taken as that function's postcondition, it is worthwhile to use expression functions rather than ordinary functions whenever possible.
2. Thanks to Rod Chapman for suggesting the name Verification Driven Development.
3. The name of the method is loosely derived from the concept of using information flow as the central tool in the design of the objects or entities making up the system. INFORMED is an acronym for **IN**formation **F**low **OR**iented **ME**thod of **D**esign.
4. For example, defining a record type implicitly defines equality and field selection operations. We might add an additional operation such as less than or equal to.
5. The Heartbleed Bug in the popular OpenSSL library was, in part, a result of combining sensitive data with nonsensitive data in the same buffer object.
6. The classic colloquial definitions: Verification – are we building the product right? Validation – Are we building the right product?
7. Bob must also believe that the various cryptographic algorithms used are secure.

9 Advanced Techniques

1. Ada assertions are executable when the assertion policy is set to Check. So when the assertion policy is set to Check, ghost functions may be executed.
2. A future version of SPARK may allow 'Old to be used in Assume pragmas.
3. Thanks to Johannes Kanig.
4. See `http://why3.lri.fr/\#provers` for a complete list of compatible provers.
5. Also keep subtype predicates in mind, although at the time of this writing `Dynamic_Predicate` is not supported by SPARK.
6. This limitation has since been partially removed, and work is ongoing to improve the SPARK tools further in this area.
7. A real implementation would need to choose the positive or negative square root depending on the value of `X`.

References

Ada Conformity Assessment Authority. 2012. *Ada Reference Manual, ISO/IEC 8652:2012 (E)*. 3rd edn.

Adams, C., Cain, P., Pinkas, D., and Zuccherato, R. 2001 (August). *RFC-3161: Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)*. Freemont, CA: Internet Engineering Task Force.

Aho, Alfred V., Lam, Monica S., Sethi, Ravi, and Ullman, Jeffrey D. 2007. *Compilers Principles, Techniques, & Tools*. 2nd edn. Boston, MA: Addison Wesley.

Amey, Peter. 2002. Correctness by Construction: Better Can Also Be Cheaper. *CrossTalk, the Journal of Defense Software Engineering*, **15**(3), 24–28.

Ammann, Paul, and Offutt, Jeff. 2008. *Introduction to Software Testing*. Cambridge: Cambridge University Press.

Barnes, John. 2012. SPARK*: The Proven Approach to High Integrity Software*. http://www.altran.co.uk, UK: Altran Praxis.

Barnes, John. 2014. *Programming in Ada 2012*. Cambridge: Cambridge University Press.

Beizer, Boris. 1990. *Software Testing Techniques*. New York: Van Nostrand Reinhold.

Ben-Ari, Mordechai. 2009. *Ada for Software Engineers*. 2nd edn. London: Springer-Verlag.

Bjørner, Nikolaj. 2012. Taking Satisfiability to the Next Level with Z3. Pages 1–8 of: Gramlich, Bernhard, Miller, Dale, and Sattler, Uli (eds), *Automated Reasoning*. Lecture Notes in Computer Science, vol. 7364. Berlin: Springer.

Black, Rex. 2007. *Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional*. Indianapolis: Wiley.

Blair, Michael, Obenski, Sally, and Bridickas, Paula. 1992. *Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia*. Tech. rept. GAO/IMTEC-92-26. Washington, DC: United States General Accounting Office.

Bobot, François, Filliâtre, Jean-Christophe, Marché, Claude, and Paskevich, Andrei. 2011. Why3: Shepherd Your Herd of Provers. In: *In Workshop on Intermediate Verication Languages* (pp. 53–64). Wroclaw, Poland.

Chapin, Peter. 2014. *Thumper*. `https://github.com/pchapin/thumper`.

Chapman, Roderick, Botcazou, Eric, and Wallenburg, Angela. 2011. SPARKSkein: A Formal and Fast Reference Implementation of Skein. Pages 16–27 of: *Proceedings* *of the 14th Brazilian Conference on Formal Methods: Foundations and Applications*. SBMF'11. Berlin: Springer-Verlag.

Chapman, Roderick, and Schanda, Florian. 2014. Are We There Yet? 20 Years of Industrial Theorem Proving with SPARK. Pages 17–26 of: Klein, Gerwin, and Gamboa, Ruben (eds), *Interactive Theorem Proving*. Lecture Notes in Computer Science, vol. 8558. Switzerland: Springer International Publishing.

Comar, Cyrille, Kanig, Johannes, and Moy, Yannick. 2012. *Integrating Formal Program Verification with Testing*. Tech. rept. AdaCore. `http://www.adacore.com/uploads_gems/Hi-Lite_ERTS-2012.pdf`.

Croxford, Martin, and Chapman, Roderick. 2005. Correctness by Construction: A Manifesto for High-Integrity Software. *CrossTalk, the Journal of Defense Software Engineering*, **18**(12), 5–8.

Dale, Nell, and McCormick, John. 2007. *Ada Plus Data Structures: An Object-Oriented Approach*. 2nd edn. Sudbury, MA: Jones and Bartlett.

Dale, Nell, Weems, Chip, and McCormick, John. 2000. *Programming and Problem Solving with Ada 95*. 2nd edn. Sudbury, MA: Jones and Bartlett.

Davis, Noopur, and Mullaney, Julia. 2003. *The Team Software Process (TSP) in Practice: A Summary of Recent Results*. Tech. rept. CMU/SEI-2003-TR-014 ESC-TR-2003-014. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

DeRemer, Frank, and Kron, Hans. 1975. Programming-in-the-Large Versus Programming-in-the-Small. Pages 114–121 of: *Proceedings of the International Conference on Reliable Software*. New York: Association for Computing Machinery.

Dross, Claire, Efstathopoulos, Pavlos, Lesens, David, Mentré, David, and Moy, Yannick. 2014. *Rail, Space, Security: Three Case Studies for* SPARK *2014*. `http://www.spark-2014.org/uploads/erts_2014.pdf`.

Dutertre, Bruno. 2014. Yices 2.2. Pages 737–744 of: Biere, Armin, and Bloem, Roderick (eds), *Computer-Aided Verification (CAV'2014)*. Lecture Notes in Computer Science, vol. 8559. Heidelberg, Germany: Springer.

Eisenstadt, Marc. 1997. My Hairiest Bug War Stories. *Communications of the ACM*, **40**(4), 30–37.

English, John. 2001. *Ada 95: The Craft of Object-Oriented Programming*. `http://www.adaic.org/resources/add_content/docs/craft/html/contents.htm`.

Epp, Susanna S. 2010. *Discrete Mathematics with Applications*. 4th edn. Pacific Grove, CA: Brooks/Cole Publishing.

Gersting, Judith. 2014. *Mathematical Structures for Computer Science*. 7th edn. New York: W.H. Freeman.

GNAT, 2015a. GNAT Reference Manual, `http://docs.adacore.com/gnat_rm-docs/html/gnat_rm/gnat_rm.html`

GNAT, 2015b. GNAT User's Guide, `http://docs.adacore.com/gnat_ugn-docs/html/gnat_ugn/gnat_ugn.html`

Hall, Anthony, and Chapman, Roderick. 2002. Correctness by Construction: Developing a Commercial Secure System. *IEEE Software*, **19**(1), 18–25.

Humphrey, Watts. 2000. *Introduction to the Team Software Process*. SEI Series in Software Engineering. Boston, MA: Addison Wesley.

Humphrey, Watts. 2004. *Security Changes Everything*. Keynote address presented at the ACM SIGAda Annual International Conference, November 14–18, Atlanta, GA.

Humphrey, Watts. 2006a (January). *Defective Software Works*. News at SEI. `http://www.sei.cmu.edu/library/abstracts/news-at-sei/wattsnew20041.cfm`.

Humphrey, Watts. 2006b (February). *Security Changes Everything*. News at SEI. `http://www.sei.cmu.edu/library/abstracts/news-at-sei/wattsnew20042.cfm`.

International Telecommunication Union. 2002 (July). *Information Technology – ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER), Distinguished Encoding Rules (DER)*. Geneva, Switzerland.

Jones, Capers. 2000. *Software Assessments, Benchmarks, and Best Practices*. Addison-Wesley Information Technology Series. Boston: Addison Wesley.

Jones, Capers. 2012 (September). *Software Quality in 2012: A Survey of the State of the Art*. Software Quality Group of New England. `http://sqgne.org/presentations/2012-13/Jones-Sep-2012.pdf`.

Jones, Capor. 2013. *Software Defect Origins and Removal Methods*. Tech. rept. Narragansett, RI: Namcook Analytics LLC.

Jorgensen, Paul. 2008. *Software Testing: A Craftsman's Approach*. 3rd edn. Boca Raton, FL: Auerbach Publications.

Kaner, Cem, Falk, Jack, and Nguyen, Hung Quoc. 1999. *Testing Computer Software*. 2nd edn. Indianapolis, IN: Wiley.

Knight, John, DeJong, Colleen, Gibbs, Matthew, and Nakano, Luis. 1997 (September). Why Are Formal Methods Not Used More Widely? In: Holloway, Michael, and Hayhurst, Kelly (eds), *Proceedings of the Fourth NASA Langley Formal Methods Workshop* pp. 1–12. Hampton, VA: NASA.

Knuth, Donald. 1998. *The Art of Computer Programming: Seminumerical Algorithms*. Vol. 2. Boston, MA: Addison-Wesley.

Mao, Wenbo. 2004. *Modern Cryptography Theory and Practice*. Upper Saddle River, N.J.: Pearson.

Marsh, William, and O'Neill, Ian. 1994. *Formal Semantics of* SPARK. Tech. rept. Bath, England: Program Validation (available from Altran Praxis).

McCormick, John. 1997. Forum Letter. *Communications of the ACM*, **40**(8), 30.

McCormick, John W., Singhoff, Frank, and Hugues, Jérôme. 2011. *Building Parallel, Embedded, and Real-Time Applicatins with Ada*. Cambridge, England: Cambridge University Press.

Mills, Harlan, Dyer, Michael, and Linger, Richard. 1987. Cleanroom Software Engineering. *IEEE Software*, **4**(5), 19–25.

Moy, Yannick, Ledinot, Emmanuel, Delseny, Herve, Wiels, Virginie, and Monate, Benjamin. 2013. Testing or Formal Verification: DO-178C Alternatives and Industrial Experience. *IEEE Software*, **30**(3), 50–57.

NASA. 2011 (January). *National Highway Traffic Safety Administration Toyota Unintended Acceleration Investigation*. Technical Assessment Report TI-10-00618. Washington, DC: NASA Engineering and Safety Center.

New York University. 2014. *CVC4: The SMT Solver*. `http://cvc4.cs.nyu.edu/web/`.

National Institute of Standards and Technology. 2002 (May). *The Economic Impacts of Inadequate Infrastructure for Software Testing*. Planning Report 02-3. Washington, DC: NIST.

OCamlPro. 2014. *The Alt-Ergo Theorem Prover*. `http://alt-ergo.lri.fr/`.

OpenSSL Project. 2014a. *OpenSSL Cryptography and SSL/TLS Toolkit*. `https://www.openssl.org/`.

OpenSSL Project. 2014b. *OpenSSL Vulnerabilities*. `https://www.openssl.org/news/vulnerabilities.html`.

Parnas, David Lorge, and Madey. 1995 (January). Functional Documents for Computer Systems. *Science of Computer Programming*, **25**(1), 41–61.

Pattis, Richard E. 1988. Textbook Errors in Binary Searching. *SIGCSE Bulletin*, **20**(1), 190–194.

Paulk, Mark C. 2009. A History of the Capability Maturity Model for Software. *ASQ Software Quality Professional*, **12**(1), 5–19.

Radio Technical Commission for Aeronautics (RTCA). 2011a. *DO-178C Software Considerations in Airborne Systems and Equipment Certification*. RTCA and European Organisation for Civil Aviation Equipment (EUROCAE).

Radio Technical Commission for Aeronautics (RTCA). 2011b. *DO-333, Formal Methods Supplement to DO-178C and DO-278A*. RTCA and European Organisation for Civil Aviation Equipment (EUROCAE).

Riehle, Richard. 2003. *Ada Distilled: An Introduction to Ada Programming for Experienced Computer Programmers*. Tech. rept. Salinas, CA: AdaWorks Software Engineering.

Rosen, Kenneth. 2011. *Discrete Mathematics and Its Applications*. 7th edn. New York: McGraw-Hill.

SPARK Team. 2011 (September). *INFORMED Design Method for SPARK*. Bath, England. `http://docs.adacore.com/sparkdocs-docs/Informed.htm`.

SPARK Team. 2014a. SPARK *2014 Reference Manual*. New York: AdaCore. `http://docs.adacore.com/spark2014-docs/html/lrm/`.

SPARK Team. 2014b. SPARK *2014 Toolset User's Guide*. New York and Paris: AdaCore. `http://docs.adacore.com/spark2014-docs/html/ug/`.

Stallings, William. 2014. *Cryptography and Network Security, Principles and Practice*. 6th edn. Upper Saddle River, N.J: Pearson.

Wikibooks. 2014. *Ada Programming*. `http://en.wikibooks.org/wiki/Ada_Programming`.

Index