Lovelace: An Ada 95 Tutorial

David A. Wheeler

Institute for Defense Analyses[1]
1801 N. Beauregard St.
Alexandria, VA 22311-1772
(703) 845-6662
wheeler@ida.org

Lovelace is a free interactive tutorial for Ada 95 [RM 95] and is available through the World Wide Web (WWW). This paper describes Lovelace, including basic facts about it, its organization, implementation, user comments, and lessons learned. This paper should be of use to educators considering the use of Lovelace in courses using Ada, users of Lovelace, and educators considering the development of computer-aided instruction programs for the World Wide Web (WWW).

1. Basic Facts About Lovelace

1.1 Lovelace's Purpose and Expected Users

The purpose of Lovelace is to teach the Ada 95 computer programming language, with sufficient coverage so the user can become a beginning Ada programmer. Lovelace is intended for software developers who know another algorithmic programming language (such as C, C++, Fortran, or Pascal) but who do not know Ada.

1.2 Current Status

The current version of Lovelace as of this writing is version 4.3. This version has 68 sections grouped into 17 lessons. Its text includes more than 54,000 words and it is stored in 273 HTML files. Lovelace would occupy more than 300 pages if printed.

1.3 How to Use Lovelace

Internet access is not required to use Lovelace, but a program called a "web browser" is required. Two popular web browsers are Mosaic and Netscape, but others (such as lynx) work as well.

Once you have a web browser, there are two ways to use Lovelace:

a. An Internet connection. To do this, "open" (using a web browser) the following URL: "http://lglwww.epfl.ch/Ada/Lovelace/lovelace.html".

---

1 The publication of this paper does not indicate endorsement by the U.S. Department of Defense (DoD) nor the Institute for Defense Analyses (IDA), nor should the contents be construed as reflecting the official positions of those organizations.

b. A local connection. To use Lovelace locally, you must somehow download Lovelace to a place where your computer can read it. One simple approach is to buy it on a CD-ROM [Walnut 95]. Another approach is to download Lovelace; instructions on how to download Lovelace are available at the main Lovelace site, or you can download Lovelace from the Public Ada Library (PAL) at "ftp://wuarchive.wustl.edu/languages/ada/".

2. Lovelace Organization

2.1 Overall Organization

Lovelace consists of a home page, the main tutorial text, and supporting pages. The main tutorial text is organized so that the material is divided into small, easily-understood units of information. The material is broken into a set of lessons (currently numbered 1 to 17), which are further broken into sections (numbered 1.1, 1.2, and so on). Each section is short (about one to one-and-a-half pages of text), and each section explains only a few key concepts. The supporting pages provide related material, for example, the bibliography. Figure 1 illustrates this organization (the arcs indicate some primary hypertext links, but not all hypertext links are shown).

Most sections end with a single-question quiz. The purpose of each quiz is to ensure that the user has grasped at least some of the principles in that section. The user may skip the quizzes, but the quizzes are recommended because the process of taking each quiz helps the user remember the key points. The quizzes often turned out to be the most difficult part of a section to create: they need to ask the user some key point while being neither too easy nor too difficult.

Most current WWW tutorials don't include quizzes and are merely static, non-interactive books. Lovelace, by contrast, is much more interactive and uses the quizzes to help the user remember the material through reinforcement. There's no penalty for wrong answers, so there's no disincentive for quiz use. Some users may not want to take a quiz, so users can skip quizzes and move on to the next section.

Since the intended user already knows some other algorithmic programming language, many subjects are glossed over or assumed (e.g. variables, the semantics of assignment, the meaning of expressions). Examples of similar capabilities in Pascal, C, and C++ are included because many of the users will already be familiar with those languages. A different kind of tutorial would be needed for those without such knowledge (e.g. see the LAW tutorial in the "Related Tutorials" section).

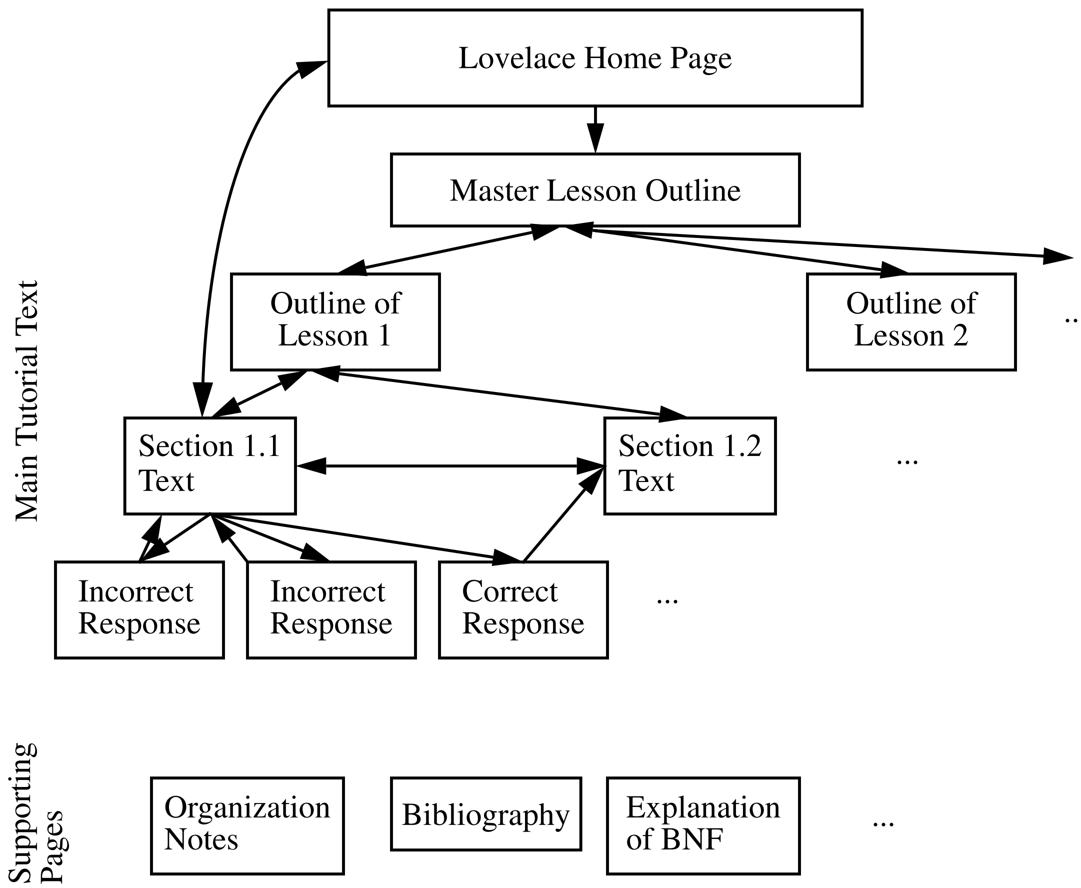

Figure 1. Lovelace's Overall Organization

Lovelace is designed to have "zero start-up cost". By intent, users only need a WWW browser (such as Mosaic or Netscape)—there's no requirement to install a compiler or any other tool. Thus, people who are just "Internet surfing" could simply start using this tutorial immediately. Naturally, people will learn best if they actually develop their own programs, but that's not required by Lovelace.

2.2 Organization of the Main Tutorial Text

Lesson organization is always a tricky issue for a tutorial. Lovelace starts "top down" for a short time so that users can quickly get the highlights, but then switches to a "bottom up" approach. The reason for switching to a "bottom up" approach is because it is very difficult to explain higher-level organization issues without first explaining the basic terms. Lovelace was designed to attempt to take advantage of both a "top down" and "bottom up" organizational approach. Table 1 lists the current lessons in Lovelace.

Table 1. Lovelace Lessons

Lesson Number	Lesson Title
1	Brief Introduction to Ada.
2	Basic Ada Structure (Packages).
3	Ada Lexical Elements.
4	Procedures and Type Integer.
5	Statements (if, loop).
6	Basic Types (Float, Boolean, subtypes, record).
7	Object-Oriented Programming.
8	Introduction to String Types.
9	Basic Input/Output.
10	Exceptions.
11	Generics.
12	Access Types.
13	Tasks and Protected Types.
14	Ada-related Information.
15	Ada Program Structure.
16	Interfacing to Other Languages.
17	Miscellaneous Ada Subjects.

Lesson 1 gives an overall description of Ada and how Ada was developed. Lesson 2 describes how Ada programs are structured (using packages) and gives a top-down view of the language. However, a top-down description becomes difficult to continue because so many simple terms and concepts have not been discussed. Lessons 3 through 6 thus take a bottom-up approach, describing basic programming capabilities of Ada 95. Lesson 7 introduces object-oriented (OO) features; the OO features are discussed before strings and input-output because many people are interested in learning specifically about them. Lessons 8 and 9 then describe strings and input-output. By the end of lesson 9, beginners should know enough to write simple interactive text programs.

Lessons 10 through 13 then introduce some of the more advanced features of Ada (exceptions, generics, access types, and tasking). The end of lesson 13 marks the end of the "bottom up" view of Ada, and the lessons after lesson 13 discuss various important topics.

Lesson 14 describes related available information (especially on-line sources). Lesson 15 completes the "top-down" view of Ada (that was started earlier in lesson 2). The "top-down" view could now be completed because the user has learned the information in the "bottom up" sections.

Lesson 16 and 17 then cover special topics that are likely to be useful to most users. Lesson 16 covers interfacing to other languages, and since C is a likely "other language", lesson 16 concentrates on interfacing Ada programs with C programs. Lesson 17 covers a range of miscellaneous subjects: language-defined attributes, efficiency, and software safety. Software safety is not a topic specific to Ada, but since many safety-critical systems are built using Ada, it seemed appropriate to specifically address safety and point to related information (as well as briefly introducing Ada-specific approaches).

2.3 Supporting Page Organization

The supporting pages were created as needed to provide additional material that did not belong in the main body of the tutorial. Hypertext links make it easy for users to get to this material, and many of the supporting pages are referenced from the tutorial home page. Some examples of this kind of material are:

- Bibliography: Lovelace references several documents, particularly in the sections on efficiency and safety. Selecting a reference moves the user to the location in the bibliography with more information about that reference.
- Organization Notes: Notes on the organization of the tutorial itself are included here.
- Explanation of BNF: Many sections include a definition in Backus-Naur Form (BNF); this page explains how to read BNF for users who do not know how to read BNF descriptions.

3. Lovelace Implementation

The standard document format for the World Wide Web is the Hypertext Markup Language (HTML). Unfortunately, HTML is a little difficult to use directly when developing a tutorial. Tutorials typically include a large number of links that repeat the same pattern, and inserting these links "by hand" is very time-consuming. For example, for each section there should be a link to the previous section, a link to the next section, an optional quiz (with separate pages for each possible answer), and a link "up" to an outline. For each quiz response, there should be a link back to the question and a link forward to the next section. Every lesson outline should have a link to the next lesson, the previous lesson, and up to the outline of all lessons. Therefore, to make tutorial generation easier, a program called "mklesson" was developed that accepted a special input format (called the "les" format) and generated the lesson files in HTML format.

Figure 2 shows how "mklesson" works. It takes as input a file representing one lesson (in les format) and a "template" file which guides what mklesson will generate. Mklesson then generates each section and a file for each possible quiz answer, as well as all the links between them and an outline of that lesson.

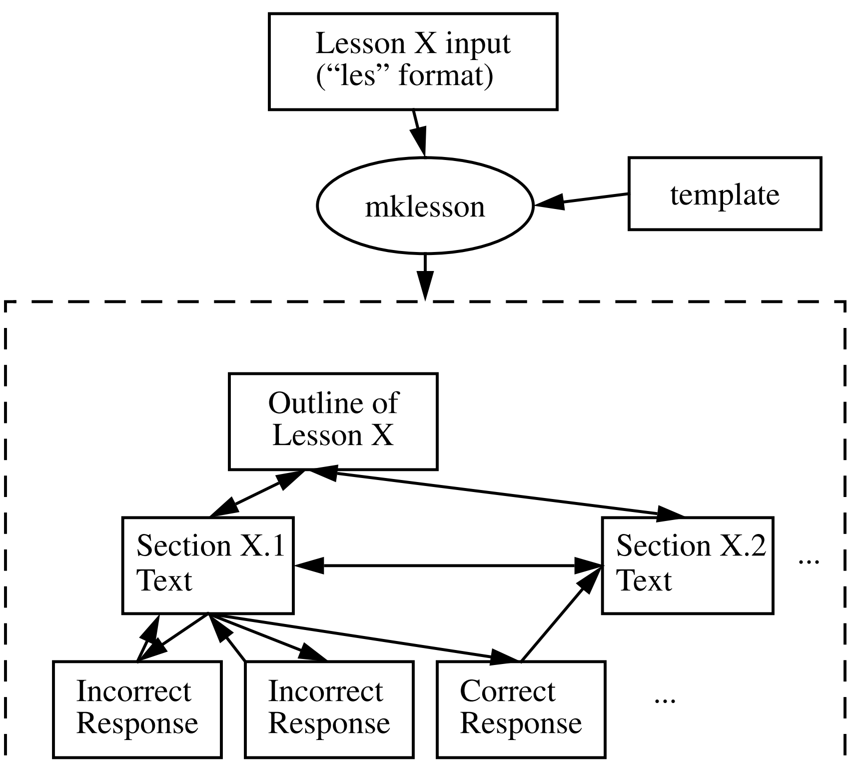

Figure 2. How mklesson Works

This approach to implementing Lovelace not only simplifies lesson creation and content modification, but it also makes it easier to change formatting later. For example, the first version of Lovelace had no graphical icons. Graphical icons (indicating the next item, previous item, and so on) were easily added by simply modifying the “template” file and rerunning mklesson.

Mklesson was designed to support other capabilities that have not been used: distribution of a single tutorial across multiple sites (and maintained by multiple people) and support of different (human) languages. Perhaps, when these capabilities are used, more can be said about them.

4. User Comments

I have depended on voluntary user comments as the primary mechanism for feedback about Lovelace. User impressions have generally been very positive. Here are some examples:

- *I have just taken the Lovelace Ada tutorial. I thought it was very informative and easy to understand. Thank you for providing it through the WWW. (Adina Ragenovich, aragen@cs.montana.edu)*
- *I finished Lovelace about a week ago. It has left a good impression on me. I think it does an outstanding job of nurturing the interest of someone who is truly interested in Ada 95. I noticed that you described it as a whirlwind tour. I think that is true, but it was a whirlwind tour with many opportunities for brief reflection and with directions for detailed inquiries. Specifically, I think that the quizzes with comments for both the right and the wrong answers really helps a novice, like me, to get some sense of progress. Of course, more can be learned by working through full blown exercises but I think it might break the momentum of someone who is curious about Ada 95 in a self-motivated way. I also thought it was a good idea to include pointers to other sources of information about Ada. (Ted Mansueto, mansueto@iastate.edu)*
- *What I appreciated most was the small, uncomplicated sample programs you used. So many books use examples that are three pages long and try to incorporated three or four new ideas. (Lisa Mae Bronkema, LisaMae5@aol.com)*
- *I just completed your Lovelace Ada Tutorial. I thought it was very well done. The lessons are broken up and ordered very well. I could do a couple of lessons each morning before getting on to my other work. The sections of each lesson are small enough that even if I got interrupted in the middle of a lesson it was only a short*

distance back to recover at a logical break.

I especially liked the end-of-section quizzes. They tended to ensure that I absorbed what I'd read and hadn't just scanned it without engaging my brain (something that's easy to do when reading long technical documents). Even the sections with simple questions kept me awake, so here's an area for improvement: Add a Quiz to the sections that don't have them, even if it's just a regurgitation of what was presented a couple of paragraphs above.

The hypertext links are also a good inclusion, I only followed a few but may come back for some more in the future. (Chip Patton, crpatton@ingr.com)

5. Related Tutorials

There are other Ada tutorials; the key on-line tutorials are:

a. Learn Ada on the Web (LAW) by Dr. Fintan Culwin. This tutorial concentrates on those who know little about software development, rather than helping programmers who already know other programming languages. LAW is available at "http://www.scism.sbu.ac.uk/law/lawhp.html".

b. Hot Ada is a hypertext self-directed Ada tutorial for PCs or Apple Macintosh from Stage Harbor Software available for a nominal fee. Hot Ada spends more time describing object-orientation in general (as well as Ada specifically), and uses a special graphical notation [Crawford 95].

c. AdaTutr is a self-directed shareware Ada tutorial available on the Public Ada Library (PAL). There is a new version for Ada 95, and an older one for Ada 83. AdaTutr includes a series of programming exercises, which are good for reinforcement but also require more time to get through the material. These courses are available at the PAL at "http://wuarchive.wustl.edu:80/languages/ada/crsware/".

The *Catalog of Resources for Education in Ada and Software Engineering* (CREASE) lists institutions and organizations that provide Ada training, as well as other material relevant to learning Ada. This is available through the Ada Information Clearinghouse at "ftp://sw-eng.falls-church.va.us/public/AdaIC/ed-train/crease95". Note that Lovelace is not listed in the CREASE, since Lovelace became available after the CREASE survey was performed.

6. Lessons Learned

The following lessons were learned in developing this tutorial:

a. Use document generators to develop a WWW tutorial. The HTML language is simply too low-level to develop a tutorial; using a program (mklesson) proved to be very helpful.

b. “Marketing” takes a long time. It took more than ten hours to identify locations where there would be an interest in Lovelace and to send information in a form useful to them. Lovelace is referenced in the general WWW directory YAHOO[2], the GNAT Ada compiler notes, and the AdaIC bulletin board. It resides in the “Home of the Brave Ada Programmers”[3] and “Public Ada Library”. It was announced in the newsgroups comp.lang.ada and comp.edu, NCSA’s “What’s New”, and the Ada Newsbites (put together by the AdaIC), but these announcements are missed by many. Lovelace is referenced or has been announced in many other places as well.

c. Use programs to detect errors. The Ada language philosophy is to use computers to catch errors, and that philosophy should also be used to develop tutorials. I originally did not use many programs to catch errors and many problems slipped through. Now, before distribution, Lovelace goes through checking by weblint (to detect HTML formatting problems), hspell (a spelling checker), GNAT (an Ada compiler, to detect Ada syntax errors), and the mklesson program itself checks for some errors.

d. Programmed instruction works well. Many users appreciated the questions at the end of each section, and believed (as I did) that the quizzes helped them to understand the material.

e. Few volunteers will actually produce a product. A number of people, with good intentions, volunteered to write a lesson or translate Lovelace into another language. Few, if any, actually produced anything. It was important to set deadlines on volunteers, and move on if they did not produce anything.

f. Web browsers have portability problems. Web browsers vary in how they display information, so test documents using many different web browsers.

---

[2] YAHOO is at URL “http://www.yahoo.com”.

[3] The Home of the Brave Ada Programmers is at URL “http://lglwww.epfl.ch/Ada/”.

7. Conclusions

Lovelace appears to be a success; users are using it and they say that it helps them understand the subject matter (the Ada 95 programming language). Many users have commented that the interactive format (presenting a short amount of material and then reinforcing this with a quiz) was particularly helpful, and it is hoped that other WWW tutorials will emulate this approach.

8. References

[Crawford 95] Crawford, Bard S. “Proposed Icons for Ada 95”. *ACM Ada Letters*. Jul/Aug 1995. Volume XV, Number 4. pp 36-45.

[RM 95] Ada 95 Reference Manual (RM). Version 6.0, Intermetrics, Inc., January 1995. Revised international standard ISO/IEC 8652:1995. For an electronic copy, see URL “http://lglwww.epfl.ch/Ada/LRM/9X/rm9x/rm9x-toc.html”.

[Walnut 95] Walnut Creek CD-ROM. “Walnut Creek Ada CD-ROM”. Walnut Creek, CA. Contact (800) 786-9907, (510) 674-0783, or “info@cdrom.com”.