# Synthesizing customizable learning environments

Thulasi Ram Naidu*, Manisha Verma*, Venkatesh Choppella* and Gangadhar Chalapaka†
*IIIT Hyderabad
*Amazon Inc. Hyderabad

***Abstract*—Making the experience of e-learning more effective requires interactive and collaborative systems to be adaptive and customizable. Specialized learning systems tend to be monolithic and difficult to extend. We present an alternative approach, where we synthesize a customizable learning environment from existing tools (Trac, SVN, reST, SQLite). The system presents the student not just with content, but an immersive experience that allows both individual and group annotations, versioning of the student's work, custom querying, and a uniform markup language to store content. We report the motivation and design of such an environment. We demonstrate the use of this system and its ability to plug into other environments by showcasing a custom interactive workbook, built for teaching and learning the principles of programming.**

***Keywords*-information technology; e-learning; learning environments;annotations; FOSS; collaboration**

## I. Introduction and Motivation

**Collaborative environments for immersive learning:** Learning environments today are rapidly evolving towards the goal of providing a completely immersive experience for the learner. Collaboration is being seen as an increasingly important aspect of learning [1]. In such a collaborative environment, the student is not merely a consumer of knowledge in a knowledge base, but an active participant in a community of teachers and learners interacting with, discovering and sharing new connections in the knowledge network.

**Problem statement: building learning environments:** This paper is about how to easily synthesize a collaborative learning environment. It is not about building another learning management system from scratch. A learning environment is different from a learning *management* system, which manages the administrative *process* of learning: homework submissions, deadlines, exams, quizzes, and uploads of announcements and material from instructors. Perhaps the most widely used example of a learning management system is Moodle [2]. A learning environment, on the other hand, is more about knowledge sharing and creation and more intimately connected with the learner and learning activities and artefacts. Several learning management systems, however, do double up as learning environments. Our experience shows, however, that certain key properties, listed below, of interactive learning are absent in learning management systems.

**Motivation: essentials of a learning environment:** We motivate this work by first examining the essential capabilities that a learning environment must support:

- **support for portable content** Content is central to a learning environment. Content creators must be able to create material that is portable, ie., runs on different platforms and can be created through several tools. Content developers should be able to create content independent of the environment to the extent possible. The learning environment should be able to use the content without monopolising it.
- **writing and sharing annotations and commenting** The student should be able to not only read the material (narrative), but annotate it and also comment on the material[3], [4], [5], [6], [7]. The student should be able to also share these annotations.
- **interacting with the learning artefacts** Learning artefacts are objects that are part of the narrative. They typically include data, plots, graphs, videos, animations, examples, code (if it is programming), etc [8], [9]. A student should be able to annotate and modify these and reinterpret the result of those modifications. Eg., change the data points and see the resultant plot, change the parameters in an animation and rerun the animation, etc.
- **versioning of previous interactions** Learning is iterative. Each iteration of learning is different from all others and adds something of value. The ability of the system to remember previous versions of interactions and iterations is important for the student to trace for herself the process of learning.
- **querying** Collaborative and immersive learning around a knowledge base has the effect of growing the knowledge base [10]. The ability to query the knowledge base is important, e.g., "get me all the comments from Shiela and Neel where they talk about Fullerenes and Entropy."

**Contributions of this work:** In this paper, we present an interactive *workbook* as an example of a *synthesized* Web 2.0 learning environment. We arrived at the synthesis approach after considering several systems (see Sections II and IV for details). We found that, in general, specialized learning systems are either monolithic or difficult to extend, and when open source, are designed either for learning management (Moodle) or content management(wiki's, Drupal etc). On the other hand, we were able to synthesize the workbook from off the shelf software (SVN version control, Trac issue tracking, Firefox browser, SQLite query language, reStructuredText markup) with only minor integration effort (a few hundred lines of Python and Javascript code). Furthermore, the approach we use is

978-1-4244-7361-8/10/$26.00 © 2010 IEEE 

general; it can be adapted with any artefact with a web interface. The workbook is a demonstration of how the system interfaces with such a learning artefact, in this case, an independent programming environment.

**Workbook main features:** The workbook we present consists of a sharable narrative annotatable by a group of learners. The narrative is embellished with various interactive components. Annotations at the individual and group level allow the users of the workbook to contribute to and share a collaborative learning space. The narrative itself may be versioned by the author. Learning artefacts (like source code in the programming environment attached to the workbook) are separately editable and versioned. The editing and versioning reinforces the iterative nature of learning. A custom querying facility allows users to search for material in the narrative and the annotations and then store the queries themselves, thus customizing the search capability. A portable markup language to store content makes it easy for the author of the narrative to create documents that can be easily integrated into the workbook, but also generate HTML, print and presentation formats. Finally, a highly customizable browser interface allows one to plug in other learning artefacts and environments. The workbook interfaces with a complete programming environment (debugger and code runner). The comments and annotations interface is also implemented as a browser plugin. The key capabilities of the system are summarised in Figure 1. A screen shot of the system showing some of its capabilities is given in Figure 2.

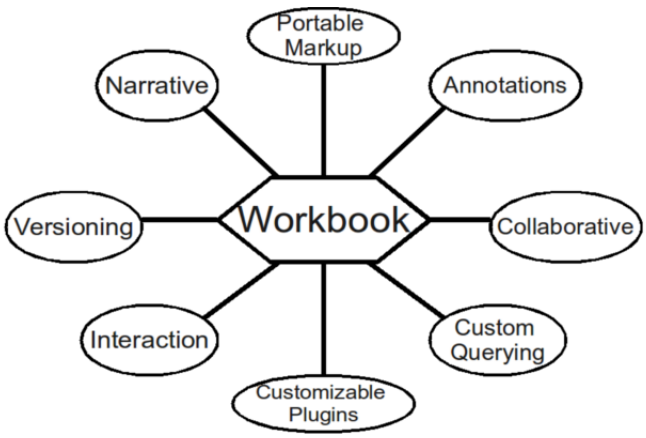


Figure 1. Key capabilities of the system.

**Paper road map:** The rest of the paper is organised as follows: In Section II we describe the approach we took in designing the system. We outline the progression of the key technology choices that we made to arrive at the current system. Next, in Section III, describe the basic architecture of the system and demonstrate how the plug-ins we built or customized interact with the architectural components of the system. In Section IV we compare our system with related work and also with existing and past systems. In Section V we suggest directions for further work, focusing on multimedia and semantic web. In Section VI we conclude by suggesting how our approach could be useful in an educational institute setting.

## II. Approach

This section is an account of how our technology choices evolved in the process of arriving at the design for the current learning environment supporting the workbook.

**Portability and web standards:** Initially, our idea was to put together a narrative for a book (not a workbook) using a simple set of HTML pages. Plain HTML, along with cascading style sheets (css), was chosen because of its extreme portability (viewable on any browser) *plus* its support for editing on a variety of platforms (text editors like vim, emacs, specialised tools like Visual studio, Dreamweaver, Bluefish etc.). There is another aspect to portability: independence of content from technology and adherence to web standards. We were keen that during the development of the content, our content is not locked into a learning management system. We considered using Moodle, the popular learning management system whose model is based on *uploading* content (typically as zip files or pdf's). Directly uploading HTML files into Moodle renders them non-relocatable (href's to relative locations do not work). This is because *Moodle is not designed to be a web container for HTML.* On the other hand, Moodle is SCORM compliant. In this paper, however, we make no pretense of portability in terms of learning objects, whose pedagogical value in some contexts is open to scrutiny [11]. Rather, our interest is in assembling components on the web, and we therefore place primacy on compatibility with web standards (HTML, CSS, Javascript).

**Versioning:** Versioning was deemed necessary from the very beginning, because of our philosophy "documents are software." A book, like software, consist of several components, usually arranged as a set of files (chapters, figures, code artefacts, tables, etc.). Furthermore the book is *built*, exactly in the sense of a software build using these components and text processing tools (like LaTeX, for example). Following the documents as software dictum, we kept HTML pages under version control (SVN). This also supported multi-author editing (akin to multiple developers on a software project). Versioning of the book becomes crucial, because book writing, like software involves endless iterations and corrections by a changing window of people, with the need to refer to older versions. Furthermore all source code artefacts in the workbook needed to be versioned, so that the student could save older code, for example.

**From HTML to resT and Sphinx:** We achieved a further increase in portability when we switched to Restructured Text (reST). While reST documents could be converted to HTML easily, they could also be converted to LaTeX, and ODT (Open Office). Also, as a markup language, reST is a lightweight alternative to HTML. (See Figure 3 for an example.) Sphinx, a related tool [12], gave us the ability to partition our narrative into separate files based on

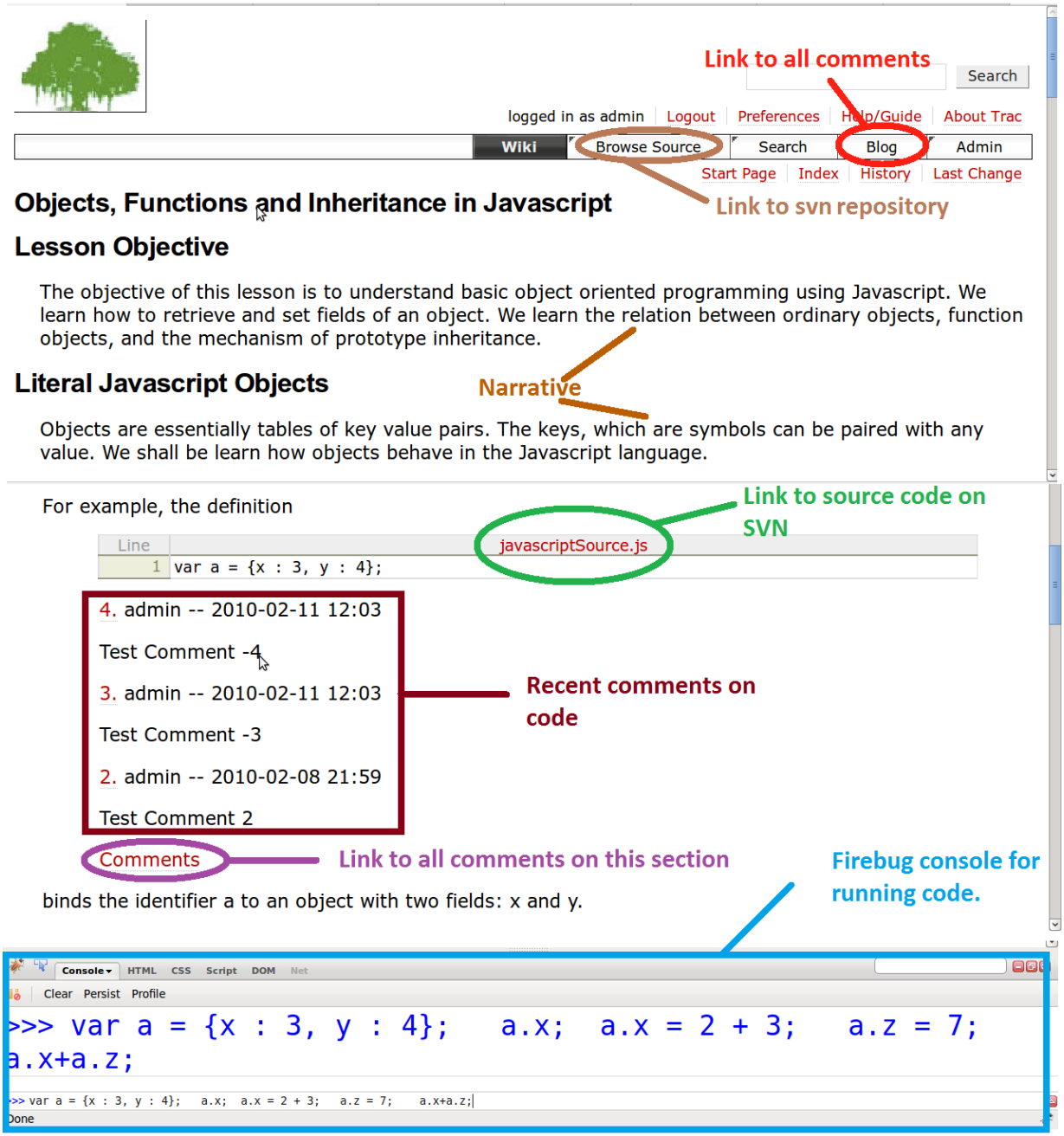


Figure 2. A screen shot of the Workbook showing integration with the programming environment (Firebug console), link to SVN, and annotations. (recent comments).

chapters and sections. Sphinx also allows searching words and phrases in the entire document.

**Browser as an interactive platform:** Progressing from a book to a workbook required that we provide support for interaction with learning artefacts. For the specific instance of the learning objective we had in mind (learning Javascript programming), we sought a programming environment that would seamlessly mesh with the browser, which was used for displaying the content. Here, Firefox became a decisive choice, because it supports an open plugin interface. The Firebug plugin was a perfect choice as a Javascript programming environment [13]. However its integration with the narrative was an issue. To click a code fragment in the book and run it in the Firebug console, we needed to write a special-purpose plugin. This plugin is a handler that transports the code fragment into the Firebug console and executes it there (details in Section III-B).

**Shared annotations:** Implementing annotations required implementing another plugin that would allow interactively associating a comment with a paragraph or a code fragment. Initially a Javascript and Ajax module was written to retrieve annotations posted by the user and store them into the SVN repository by using PHP

```
{{{
#!rst
Objects, Functions and Inheritance
==================================
.. The above is the document title
.. -*- coding: utf-8 -*-
.. role:: ci
   :class: code-inline

Literal Javascript Objects
--------------------------

Objects are essentially tables of key
value pairs.  The keys, which are
symbols can be paired with any value.
We shall be learn how objects behave
in the Javascript language.

For example, the definition
}}}
[[IncludeSource(javascriptSource.js,
          start=1, end=1)]]
[[RecentChangesBlog(recent=3, format=full,
          name=Narrator)]]
[blog:Narrator Comments]
```

Figure 3. reST markup for a section of the narrative of Figure 2.

at the backend. However using this approach for every functionality in the workbook meant fresh modules to be written either in Javascript or PHP. Hence, instead of building each module from scratch, we chose a different approach, based on synthesizing an environment from the Trac system, to which we turn our attention.

**Synthesizing a learning environment from Trac:** Trac is a popular open source project management and issue tracking tool [14]. When we investigated the suitability of Trac as a learning environment, we found that it was remarkably easy to transform it into a learning environment. This was possible only because Trac supports integration through plugins with several of the robust and portable technologies we were already using. This support allowed us to create our own plugins for implementing shared annotations and the interface to the programming environment. The integration of these tools with Trac and the overall architecture is explained in the next section.

## III. Architecture

In this section we discuss the overall architecture of our system. The first subsection briefly recalls the overall architecture of Trac and the other subsystems. The second subsection explains the structure of the plugins we wrote or adapted to synthesize a learning environment from Trac.

### A. Trac and Related components

The architecture of current system is shown in Figure 4. The system architecture consists of Trac and its coupling with other standard systems. We describe how this arrangement is useful for synthesizing a learning environment from Trac:

- *SVN*: We use SVN to version both the narrative (content) and the source code used in examples. Trac links to the SVN repository. Each user is provided with a copy of source code . This allows the user to browse and modify source code of the workbook. The user can also refer to older versions of the sources. The multi-user support has been implemented by branching a master copy of the source code. Thus each time a new user is added, a folder is created in the users name. The source code is copied to this folder and from there on any changes made to the code are versioned, stored and retrieved from the same folder. However we decided that only the author should have the permission to edit the content of the book. Allowing the users to modify content would give rise to a lot of issues that cannot be handled by our system at present and can be done in future. Presently only the author has the permission to version the content of the narrative. A user can write a custom query to fetch content and filter them based on version, time-stamp, author, etc. This feature is particularly useful in a shared environment to know who made what changes to the source base.
- *Markup*: Trac's wiki and content management supports reST markup language. This has several useful implications. Files from SVN may be displayed as reST in Trac's source browser. This allows us to maintain content *independently* in SVN, while displaying it within Trac using reST markup. Second, it allows us to preserve the ability to generate LaTeXfrom the source, independent of Trac. Comments and annotations can also freely use the reST markup syntax.
- *Querying*: Trac comes equipped with querying interface to the SqLite relational database system. This interface can be used for several types of queries in the system, like gathering recent comments (public and )about a topic, searching the narrative, and analytics, for example. Custom queries may be created and stored. Figure 5 shows the query used to generate the three most recent annotations.
- *Customization Using Plugins*: Finally, the easily customizable architecture of Trac is key to transforming it from a project management tool to a learning environment. The effort involved in this transformation was writing a few plugins and customizing two other existing plugins. These plugins and their interfaces with the Trac system are discussed in Section III-B.

**Portable development and content hosting using Trac:** Fortunately, Trac allows us to stay faithful to our goal of independent and portable content development. At the same time, it works as a container for the narrative. It supports interaction and annotations via the plugins. To use Trac to achieve the separation of content and hosting, we outline the following process: content is developed

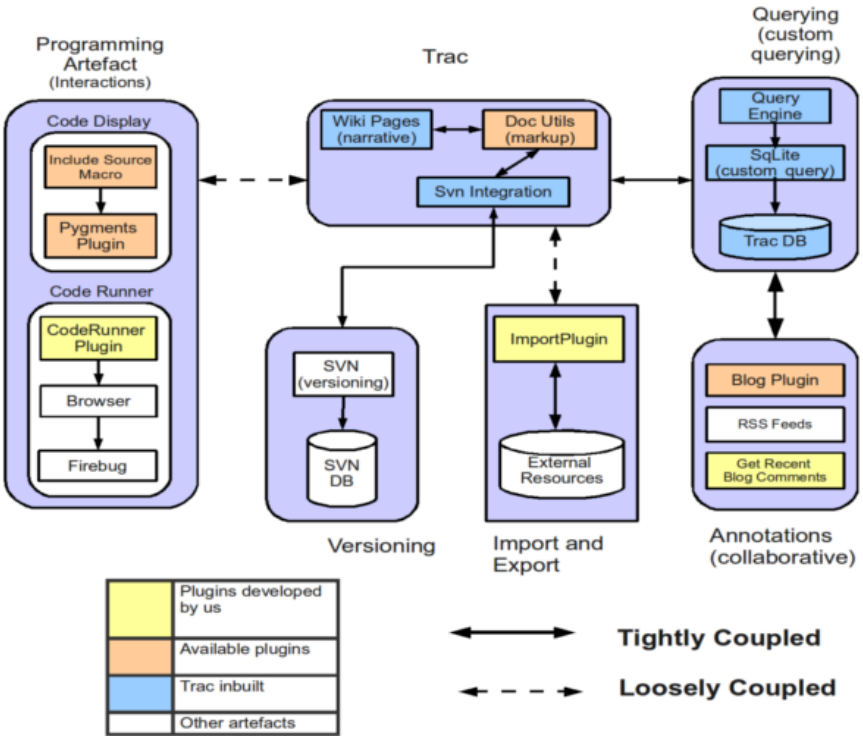


Figure 4. Architecture of the system. Components tightly (respectively loosely) coupled to Trac support (respectively extend) the basic functionality of Trac.

```
# Select name of post,comment,timestamp
#  of three most recent  annotations.

SELECT name of post, comment, time
FROM   fullblog_comments
WHERE  name="Narrator"
  ORDER BY number DESC LIMIT 3;
```

Figure 5. SQLite query to retrieve three most recent annotations for the section. This query is invoked by the RecentBlogComments macro (embedded in the narrative markup of Figure 3.)

independent of Trac in the reST format using Sphinx. Then the content is imported into Trac using a special purpose script. Trac's in built support for reST makes this possible. Once the narrative is integrated into Trac, users can annotate it . These annotations do not affect the narrative sources. Also, source code artefacts are developed independent of both the narrative and Trac. However, once developed they integrate into Trac via Trac's SVN plugin described in III-B.

### B. Plugins and their interface to Trac

Many of the features that we outlined above do not come with the default installation of Trac. Various plugins that we used, wrote, configured or installed on top of Trac are described below:

**Plugins for supporting programming artefacts:** The *includeSourceMacro* [15] plugin loads the user's version of source code from files in SVN into Trac so that source code can be included into the narrative. Various options are available to customize the inclusion:line numbers, range, version number etc. The *Pygments* plugin [16] supports colouring of code displayed in the narrative. Pygments is a general purpose plugin used for highlighting code of several programming languages in Python based document generation tools like Sphinx. The *Firebug* plugin provides a javascript programming environment tightly integrated with the Firefox browser, which allows a user to run, edit and debug javascript code. The *CodeRunnerPlugin* plugin acts as a link between the source code in the narrative window and the Firebug window. It ships the code to the Firebug console and runs it. This is one of the two plugins we had to write from scratch. It was about 100 lines of Javascript and Python code.

**Plugins to support narrative and source code interaction:** The *Wiki Pages* component forms the presentation layer of the learning environment and interacts with the user. The wiki displays the narrative and content that is stored in the Trac database. The *Docutils* plugin [17] allows the Trac wiki to use reST as its default markup language. The *Querying engine* is a component internal to Trac. It enables the user to write custom SQLite queries on the SVN repository or Trac database. The *Svn-Trac* plugin [18] integrates Trac with the SVN repository. The plugin requires some minor configuration after Trac installation.

**Annotations and Querying:** To implement shared annotations and commenting, we adapted *Blog-plugin* [19], an already available Trac plugin. The plugin supports personal and group comments. Every section in narrative is associated with a new post in the blog. A link at

the end of every section shows all comments on that section. Additionally, *GetRecentBlogComments*, a plugin which we developed, shows the most recent annotations on that section. The plugins support various filters to help customize the annotations, including the number of annotations, time limits between which comments are to be shown, and the author of the comments. The plugins are easy to build, thanks to SQLite. *GetRecentBlogComments* is actually implemented as a macro that translates into an SQL query. (See Figure 5.)

## IV. Related Systems

Collaborative learning environments have been an active area of research since the 1990's. Today there are several ebooks, workbooks, systems and learning management systems that support various learning environment features. We categorise these related works broadly as web-based e-books, annotation based systems, learning management systems, wiki and related systems, and others. We compare these with the attributes of learning environments outlined in Section I. Our survey is slanted towards learning environments for information technology and programming education. A summary of our comparison with the more prominent and current systems is shown in Table I.

**e-Books:** Web based e-books have been popular since the beginning of the web. Earlier books were in HTML and read-only. Amongst the most popular is the reference site w3schools.com [21]. It supports editing/running of code present in the narrative, but not annotations, versioning or querying. Eloquent Javascript is an online book on javascript programming that supports execution and modification of code examples [8]. The O'Reilly publication *Real World Haskell* supports user comments after every paragraph, but is not interactive [20]. Sage, an online workbook for mathematics education is the closest to our idea of an learning environment [9]. Sage supports the notion of a notebook within which one can create, collaborate, and publish interactive worksheets that also include Python code scripts. However, there is no provision for versioning or annotations in one's workspace. Several websites also support the feature of content uploading by the authors [24]. These websites can be used by authors to publish and extend content in modules or lectures but these sites lack the feature of personalized learning environment as the content can niether be annotated nor discussed amongst peer groups.

**Annotation-based systems:** Several early systems incorporated group annotations. ComMentor [3], CoNote [4], VirtualNotes [5], GroupWeb [6], and the system of Shickler et al [7] support personal and shared annotations. Some of these systems, however, require separate and specialised browsers to be installed, which limits their use as a generic user interface. Besides they also lack support for narrative, learning artefacts, versioning of text, custom querying and use of project management tools. Modern pdf viewers allow the reader to annotate and save annotations, but without any sharing or querying. The role of shared annotations in improving collaboration and interaction between peers and learning has been recently empirically studied by Lan and Jiang[25].

**Systems with query support:** An early example of a system that supports querying of annotations is the web-based e-book reading system of Chen and Wang [10]. The students can query related knowledge: the hypertext technique and query strings are used to provide them with adequate reference information. It also forecasts the reading performance of the user based on his/her interaction with the content. However, they do not consider the problem of integrating learning artefacts, or versioning.

**Learning management systems:** Moodle is perhaps the most widely used internet based learning management system [2] today. It is open source and highly customizable. However, it isn't designed to work as a learning environment that involves user interaction with artefacts like a programming workbook, for example. It has no notion of a narrative; non-editable material is typically uploaded by instructors as files in pdf or doc format. There is support for a wiki, and discussion groups, but there is no way to link to the posted material. Moodle functions as a repository for learning objects. It supports collaboration in specific ways : there is a database for lectures; tutorials and quizzes, there are discussion groups for students and teachers. However, Moodle is not connected to a versioning system at the back end. Recently, a module to support annotations in Moodle has been proposed [23].

**Wiki's and content management systems:** Mediawiki is a popular content management system, but we don't find it suitable for teaching programming (no interaction, saving and versioning of source code, no paragraph based commenting mechanism) [22]. Mediawiki does not currently integrate with SVN. Also once the content is on the wiki the author loses control over portability of the narrative. Also, at this point, Trac has better support than Mediawiki for integrating interactive artefacts like programming environments. There isn't any feature of editing and running of the code on a wiki.

## V. Future Work

**Field testing:** The synthesised learning environment we demonstrate in this paper is currently a research prototype and a testbed for further experimentation with learning environments. The system is currently web hosted on an experimental basis [26]. We plan to field test a version of it in a full classroom setting later this year. Meanwhile, there are several technical and process level issues that we are working out: generate the content for the programming book, scaling the number of users, hosting on a reliable platform, etc. Additional plugins to automatically insert macro invocations (like RecentBlogComments) into the narrative and simplify the markup in other ways still need to written.

**Multimedia:** The learning platform itself is currently missing any multimedia integration (videos, audio clips, etc.), and also animation. The relatively easy effort in

Table I
COMPARISON OF EXISTING SYSTEMS

| | Example Code | Annotations | Language | Open source | Extensible/Customizable |
|---|---|---|---|---|---|
| EJ[8] | Run/Edit | No | HTML | Yes | No |
| RWH[20] | No | Shared | HTML | No | No |
| SAGE[9] | Run/Edit | Discussion | reST | Yes | Yes |
| W3Schools[21] | Run/Edit | No | HTML | Yes | No |
| Media Wiki[22] | No | Discussion | Wiki Markup | Yes | Yes |
| Moodle[2] | No | Personal/Shared(.ppt)[23] | Wiki Markup | Yes | Yes |
| Acrobat | No | Personal | - | No | No |
| Our System | Run/Edit | Shared/Personal | reST | Yes | Yes |

putting together the system encourages us to believe that such a learning environment could be used by teachers and students in other disciplines, like humanities, where content is more dominant than interaction. Another challenge is to use existing content available in different formats like pdf and integrate them into the learning environment (with support for annotations, etc.). For example, Javascript hooks in pdf could be exploited do this integration. This could allow personalized annotations in pdf files to integrate with Trac.

**Integrating learning with semantic web:** Integration of semantic web with our system is a natural and promising direction for further work. Semantic web technologies for information that aid students in learning [27]. Integration of e-learning systems and Semantic web for semantic services like semantic browsing, semantic search or smart question answering with the system are interesting possibilities for the future [28]. Engineering these on top of the current Trac based environment will also benefit project management, Trac's original application domain.

## VI. CONCLUSION

We have demonstrated how a learning environment can be synthesized using "off-the-shelf" open source tools. We have also shown the value of this approach: in terms of adaptability and flexibility. The system we have built also allows integrating in artefacts of learning, like an entire environment for running program code. The relative ease with which we could put together such a system is a vindication of our approach to leverage open standards (like www, web 2.0) and highly customizable and already popular open source systems. The adherence to web standards allowed us to synthesize the system out of other web based systems.

The technology landscape today opens up a new type of innovation model. In this model people recombine existing systems in interesting and useful ways, rather than build systems *ab initio*. We feel that educational institutes should embrace such models and help spawn a new generation of *application integration engineers* who continuously explore, integrate and customize such systems for the user base of that organization. The system we have demonstrated here in this research is evidence of the potential of such an innovation model. Following the idea of introducing such models and examining its various implications is clearly an interesting topic for future study by the education community.

## VII. ACKNOWLEDGMENTS

This work was partly supported by grants from the Government of India Ministry of Information Technology. We are grateful for the systems support provided by the the EnhanceEdu laboratory of IIIT Hyderabad in carrying out this project.

## REFERENCES

[1] N. A. Mukti, M. Dayana Razali, F. Ramli, H. B. Zaman, and A. Ahmad, "Hybrid learning and online collaborative enhance students' performance," in *Proceedings of the Fifth IEEE International Conference on Advanced Learning Technologies*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 481 – 483.

[2] M. Dougiamas, "Moodle," 2002. [Online]. Available: http://moodle.org/

[3] M. Röscheisen, C. Mogensen, and T. Winograd, "Beyond browsing: shared comments, soaps, trails, and on-line communities," *Comput. Netw. ISDN Syst.*, vol. 27, no. 6, pp. 739–749, 1995.

[4] J. R. Davis and D. P. Huttenlocher, "Shared annotation for cooperative learning," in *CSCL '95: The first international conference on Computer support for collaborative learning*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1995, pp. 84–88.

[5] A. Geyer-Schulz, S. Koch, and G. Schneider, "Virtual notes: Annotations on the www for learning environments," in *Proceedings of the Fifth Americas Conference on Information Systems (AMCIS)*, 1999.

[6] S. Greenberg, "Collaborative interfaces for the web," in *Human Factors and Web Development*, 1997, pp. 241–254.

[7] M. A. Schickler, M. S. Mazer, and C. Brooks, "Pan-browser support for annotations and other meta-information on the world wide web," in *Proceedings of the fifth international World Wide Web conference on Computer networks and ISDN systems*. Amsterdam, The Netherlands, The Netherlands: Elsevier Science Publishers B. V., 1996, pp. 1063–1074.

[8] M. Haverbeke, *Eloquent Javascript*. published on the web, 2006. [Online]. Available: http://eloquentjavascript.net/

[9] "Sage (mathematics software)," 2005. [Online]. Available: http://www.sagemath.org/

[10] C.-Y. Wang and G.-D. Chen, "Extending e-books with annotation, online support and assessment mechanisms to increase efficiency of learning," *SIGCSE Bull.*, vol. 36, no. 3, pp. 132–136, 2004.

[11] R. Godwin-Jones, "Emerging technologies: learning objects: scorn or SCORM?" *Language Learning and Technology*, vol. 8, no. 2, pp. 7–12, May 2004.

[12] "Sphinx-python documentation tool." [Online]. Available: http://sphinx.pocoo.org/

[13] "Firebug-an addon for firefox." [Online]. Available: https://addons.mozilla.org/en-US/firefox/addon/1843

[14] "Trac." [Online]. Available: http://trac.edgewall.org/

[15] "Includesourcemacro." [Online]. Available: http://trac-hacks.org/wiki/IncludeSourcePartialPlugin

[16] "Pygments." [Online]. Available: http://pygments.org/

[17] "Docutils." [Online]. Available: http://docutils.sourceforge.net/

[18] "Trac-svn plugin." [Online]. Available: http://trac.edgewall.org/wiki/TracSubversion

[19] "Full blog." [Online]. Available: http://trac-hacks.org/wiki/FullBlogPlugin

[20] B. O'Sullivan, D. Stewart, and J. Goerzen, *Real World Haskell*. published on the web, 2008. [Online]. Available: http://book.realworldhaskell.org/read/

[21] R. Data, *W3Schools.com*. published on the web, 1999. [Online]. Available: http://www.w3schools.com

[22] "Mediawiki," 2002. [Online]. Available: http://www.mediawiki.org/

[23] H.-T. Lin, C.-H. Wang, C.-F. Lin, and S.-M. Yuan, "Annotating learning materials on moodle lms," in *Computer Technology and Development, 2009. ICCTD '09. International Conference on*, vol. 2, Nov. 2009, pp. 455–459.

[24] "Connexions." [Online]. Available: http://cnx.org/

[25] Y.-F. Lan and Y.-C. Jiang, "Using instant messaging and annotation services to improve undergraduate programming courses in web-based collaborative learning," Aug. 2009, pp. 236 –241.

[26] "Principles of programming workbook." [Online]. Available: http://enhanceedu.iiit.ac.in/pop/trac/wiki/BookIndex

[27] J. Jovanovic, V. Devedzic, D. Gasevic, M. Hatala, T. Eap, G. Richards, and C. Brooks, "Using semantic web technologies to analyze learning content," *Internet Computing, IEEE*, vol. 11, no. 5, pp. 45 –53, Sep.-Oct. 2007.

[28] D. Kokoshi and B. Cico, "Integration of semantic web in an elearning environment," *Informatics, Balkan Conference in*, pp. 256–259, 2009.