SHARK
A Web 2.0 Service Infrastructure for Knowledge Sharing

Bo Huang, Junyong Ding, Jinquan Dai and Wenjie Zhang
Intel China Software Center, No 880, ZiXing Road, Shanghai, China

Keywords: Web 2.0, Social Networking, Knowledge Sharing, Content Model.

Abstract: Embracing interactions in computer based learning is a good approach to empowering the effectiveness of knowledge sharing, and the popularity of Web 2.0 applications signals the readiness of applying Web 2.0 technologies to computer supported education. This paper introduces *SharK* (abbreviation of *Shar*ing *K*nowledge), a Web 2.0 service infrastructure specifically designed for knowledge sharing. *SharK* adopts a novel Unified Content Model to abstract various contents inside Web 2.0 service portals, which allows easier legacy data migration, consistent and fine-grained content security control in addition to providing an extensible platform for fast new service portal construction. Besides illustrating the key design considerations, this paper also introduces three real-life *Shark*-based knowledge sharing Web 2.0 portals, which clearly demonstrate the effectiveness and efficiency of Web 2.0 portal construction based on *SharK*.

1 INTRODUCTION

Traditional education web portals are usually deployed with many education contents put onto the web so that users are able to access those contents anytime at anywhere if internet access is available. Althought those education contents are constructed in various fancy multimedia formats (video, audio, flash animation etc.), those web portals can only provide passive learning experiences, i.e. end users only need to deal with education contents pushed to them, and there is no good way for end users to provide feedback, suggestions or share knowledges, e.g. (Kesim, 2007) and (Khalifa, 2002). As a result, people are looking for better ways of constructing education web portals that make them more interactive. Fortunately, the advancement of the web technologies make this possible.

The whole internet industry grew rapidly within the past several years. The evolution is not only reflected by the significant increase of internet user number, it is also reflected by the richer representation of the contents and how those contents are generated. Since Web 2.0 applications / services expect a lot of user-generated contents by utilizing collective intelligence and social networking, they are becoming more and more popular on internet. Famous web 2.0 services providers include *Facebook*, *YouTube*, *MySpace* etc. By adopting Web 2.0 technologies such as blogging, wiki, tagging, ranking etc., some education service providers are trending well on shaping a more interactive online learning experience, e.g. (Styles, 2007), (Williams, 2005), and (Drasil, 2006).

Different Web 2.0 applications are typically designed in respectively different data models. For example, an album application might have multiple fields in which the search engine may only index the description field. While in a blog application, the data model might be totally different and the search engine usually indexes other fields such as the blog title, blog content and corresponding comments. Because of the data model variance among Web 2.0 applications, fully integrating them into a Web 2.0 service portal and making it work well requires building new data adapters for each Web 2.0 application, which is not a trivial effort. For example, the integration of Business Suite 2.0 (blog, wiki, RSS feed) took SpikeSource several quarters.

In order to make the integration of various Web 2.0 applications (both current ones and future ones) more convenient, we define a Unified Content Model in this paper, based on which we create *SharK*, a Web 2.0 service infrastructure specifically designed for knowledge sharing. As illustrated in the right side of Figure 1, data models of all Web 2.0 applications can be respectively derived from the Unified Content Model. As a result, each Web 2.0

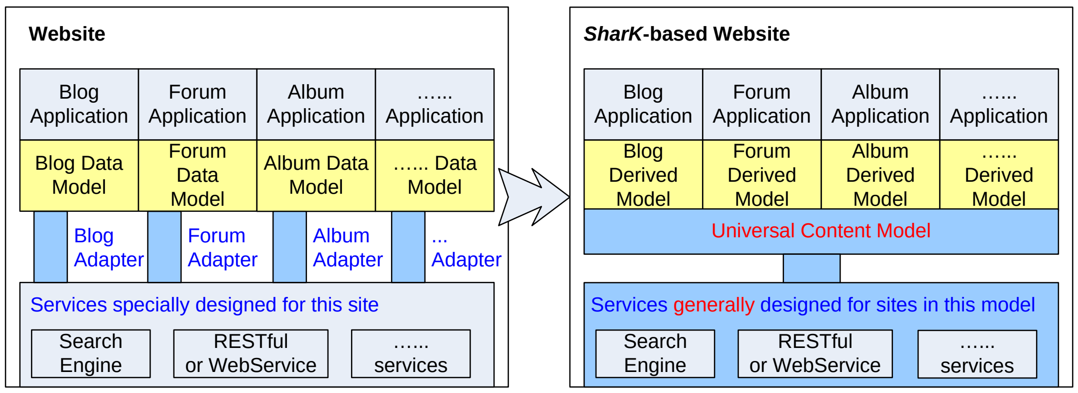


Figure 1: Comparison between traditional Web 2.0 application integration and *SharK*-based service portal construction.

application only needs to handle some application-specific data while the Unified Content Model and corresponding APIs do the rest of the work, including communicating with search engine, managing the content, processing the security settings, etc. Actually *SharK* enables fast construction of a Web 2.0 portal and the flexible integration among Web 2.0 applications developed based on the Unified Content Model. Figure 1 also gives a comparison between the integration of traditional Web 2.0 applications and *SharK*-based service portal construction.

Major contributions of this paper include the followings:

- Present a novel design of the Unified Content Model for Web 2.0 applications
- Introduce a design method to achieve UI (User Interface) separation with application logic
- Present *SharK* software architecture and how *SharK* eases the construction of Web 2.0 education/knowledge sharing portals
- Introduce three real-life *SharK*-based Web 2.0 portals that facilitate online education and knowledge sharing

The rest of this paper is organized as follows: Section 2 presents the system overview of *SharK*, followed by a detailed introduction of the design considerations in section 3. Section 4 presents three real-life *SharK*-based deployments. Section 5 introduces related work and section 6 concludes this paper.

2 SYSTEM OVERVIEW

Bearing the goal of flexible integration of Web 2.0 applications, we take modularity, extensibility and scalability into considerations when designing *SharK*. In order to reuse existing data of legacy websites, easy migration of the legacy data is also one of the design objectives.

As shown in Figure 2, the *SharK* service infrastructure can be divided into three layers:

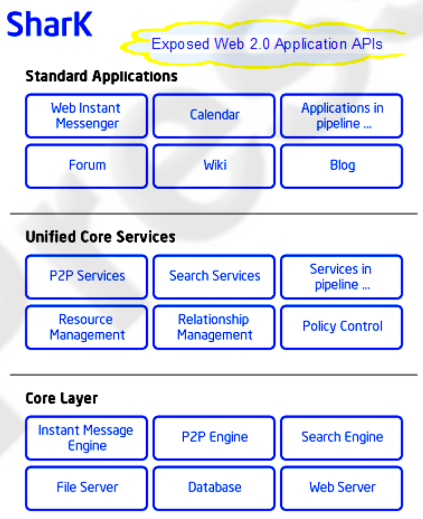


Figure 2: Software Architecture of *SharK*.

- **Core Layer**: The core layer is designed to lay out a solid foundation to support higher level layers. This layer contains major modules such as file system, database, search engine, P2P engine, web server, Instant Message engine etc. The Unified Content Model mentioned in the previous section is reflected in the data schema design of the database, which allows other modules to access the data with unified APIs. Since serving for huge volume users is one of *SharK*'s design objectives, all modules in core layers are well tuned to achieve high scalability.
- **The Layer of Unified Core Services**: This layer provides a set of APIs for conveniently building standard Web 2.0 applications. It

actually provides an abstraction of the core layer, thus minimizes the impact on the standard Web 2.0 application development when changes are made to the core layer. If more modules are added into the core layer, this layer can be easily extended by adding more core service APIs.

- **The Layer of Standard Applications:** This layer consists of typical Web 2.0 applications (Wiki, Blog, Forum, Instant Massager etc.) developed on top of unified core service APIs. *It also offers a set of standard Web 2.0 Application APIs, with which customized Web 2.0 portal can be conveniently developed.*

Clearly, the *SharK* architecture naturally fits into the knowledge sharing requirement.

- The P2P engine and corresponding P2P services make it possible to transfer/share big files such as classroom teaching video, courseware etc.
- The Instant Massager allows real-time communication among different online users, among teachers and students for example
- The efficient search engine makes it easy for users to search interested topics. Most importantly, the Unified Content Model further makes it convenient for the search engine to find results across various contents belong to different Web 2.0 applications
- Blog, Wiki and Forum bring a lot of convenience for users to express their thoughts, make comments and refine contents through tagging/ranking, which make the learning/knowledge sharing experience much more interactive

With the standard Web 2.0 application APIs offered by *SharK*, domain specific knowledge sharing portals can be easily constructed. Since those standard web applications can be freely bundled together and the UI design is well separated from the application logic (section 0), the customization effort when building a new service portal is expected to be trivial, in particular when compared with other development approaches.

3 DESIGN CONSIDERATION

Designing the whole *SharK* takes considerable efforts, and introducing the deatailed *SharK* design is out of the scope of this paper. However, this section shares several key desing considerations.

3.1 Unified Content Model

Unified Content Model is a key novelty of *SharK* design, which makes it possible to provide identical data operation interfaces for different Web 2.0 applications. The design of Unified Content Model relies on the following three basic elements:

- **Content**: Content represents a piece of information, such as text or words, a picture, a file in local file system, or an external link, etc.
- **Thread**: Several associated contents together form a thread, which is actually a session of related contents, e.g. a discussion series, a post with followed comments, etc.
- **Category**: Category is a home under which threads that have same or similar properties are put together. Category is hierarchical that can contain other categories as sub-categories

Through such abstraction, almost all data used in Web2.0 applications could be represented in this model after certain derivations (Figure 3).

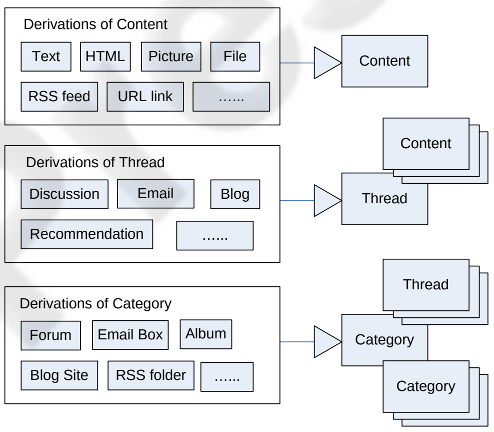


Figure 3: Content, Thread and Category.

To help better explain the Unified Content Model, Figure 4 shows the concrete design of Content, Thread and Category.

Figure 4(a) illustrates the concrete design of Content. In *Intrinsic Metadata* part, *Location* represents where the content locates, which could be a URL in internet, a path for local file system, or an ID in database. For example, we could have a specially designed format to represent the location such as *URL:http://a.com/b.html*, *PATH:D:/test.txt* or *DB:xTable/3*. This design significantly reduces the overhead brought by legacy data migration since legacy data can be either dumped as local data or use the location presentation ***points*** to it. *Visited count* is used to store how many times this content is visited.

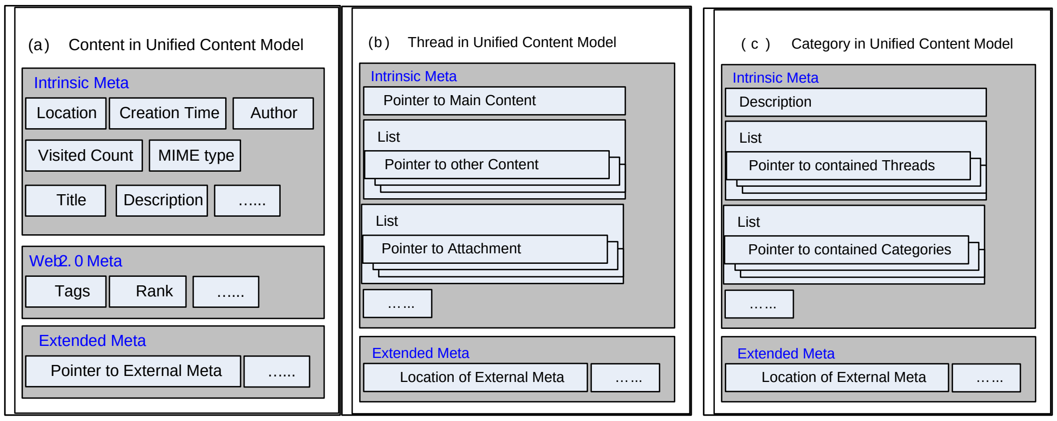


Figure 4: Concrete design of Content, Thread and Category.

Fields in *Extended Metadata* are used to represent the special properties of contents derived from the basic representation. For example, *Pointer to External Meta* contains the location of external metadata, which points to other metadata especially useful for a picture, a HTML or a file, etc. Other content fields are easy to understand. Furthermore, the hierarchical relationships among Content, Thread and Category are obvious in Figure 4.

The Unified Content Model makes it easy to define fine-grained security control over contents and bring convenience to integrate search engine into the *SharK* core layer.

- **Fine-grained Security Control**: With the Unified Content Model, security control is easy and consistent for all Web 2.0 applications. A separate security database is used to define the security settings for each piece of content, each thread and each category, no matter how these contents are used, as shown in Figure 5. Because all Standard Web 2.0 applications inside *SharK* share the same underlying infrastructure for security, Web 2.0 portal developers don't need to handle the security issue for respective Web 2.0 applications.
- **Unified Search Engine**: In the integration of traditional Web 2.0 applications (Figure 1), Web 2.0 portal developers need to spend huge effort on enabling search engine across different Web 2.0 applications. This could never be a problem in *SharK* with the adoption of the Unified Content Model. The search engine in core layer only indexes the Content and renders the search result based on the Content in the Unified Content Model. No matter which kind of Web 2.0 application is developed, its underlying data is always conformed to the Unified Data Model, thus requires no change to the search engine.

The Unified Content Model, together with other components/APIs provided by *SharK*, makes the development of a standard Web 2.0 application an easy task. Figure 6 illustrates this using the Album development as an example. As shown in the figure, developers need only take care of the design of Album-specific metadata, Album-specific application logic and UI.

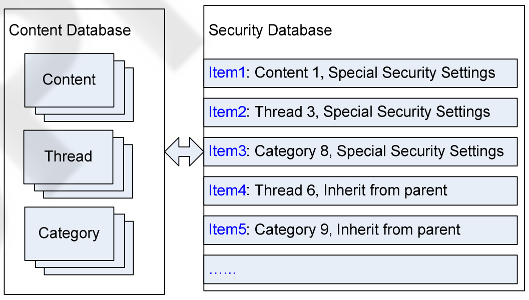

Content Database
Content
Thread
Category
Security Database
Item1: Content 1, Special Security Settings
Item2: Thread 3, Special Security Settings
Item3: Category 8, Special Security Settings
Item4: Thread 6, Inherit from parent
Item5: Category 9, Inherit from parent
……

Figure 5: Unified Security Model over Contents.

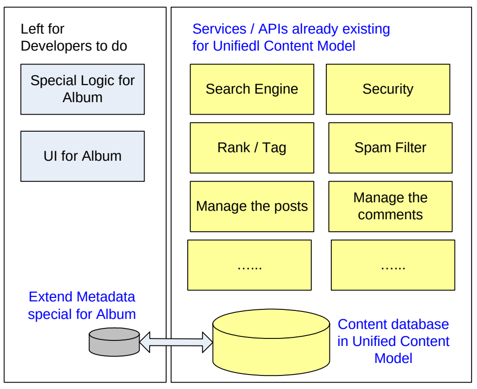

Left for Developers to do
Special Logic for Album
UI for Album
Extend Metadata special for Album
Services / APIs already existing for Unifiedl Content Model
Search Engine
Security
Rank / Tag
Spam Filter
Manage the posts
Manage the comments
……
……
Content database in Unified Content Model

Figure 6: Album development Example.

3.2 Standardizing Content Service

Content Services APIs (Figure 7) are defined on top of the Unified Content Model, which can be used to access the contents.

In those APIs, the *criteria* could be used to retrieve a list of contents/threads/categories that match certain conditions. Each criteria is actually similar to a WHERE clause in a SQL statement, e.g. "*AuthorName = 'Tom' AND VisitedCount > 10*". Implementation of those APIs will finally turn those content operations to database operations on contents/threads/categories, and *criteria* parameter will be turned to a query condition.

```
getContent(serviceURI, contentID)
getContentMeta(serviceURI, contentID)
getContentListByCriteria(serviceURI, criteria)
getThread(serviceURI, threadID)
getThreadMeta(serviceURI, threadID)
getThreadListByCriteria(serviceURI, criteria)
getCategory(serviceURI, categoryID)
getCategoryMeta(serviceURI, categoryID)
getCategoryListByCriteria(serviceURI, criteria)
……
```

Figure 7: Content Service APIs.

3.3 UI Separation

SharK takes the advantage of a template system to separate the application logic of data access with the data representation in web page, which makes the UI design much more flexible.

As illustrated in Figure 8, the controller handles the user request, invokes services and passes all essential data to the template engine. The template engine then takes over all UI tasks from the controller, picks up the suitable page (UI template) and visualizes the supplied data with it.

Therefore, UI representation in *SharK* is completely separated from the application business logic and data objects, which brings the following benefits:

- **Parallel development of UI and business logic**: Without the dependencies on underlying layer, UI developers could independently design and implement the user interface in parallel with developers constructing the application functionalities. The integration effort of UI and business logic is also significantly reduced. In some extreme cases, UI representation of standard Web 2.0 applications could be directly applied to the newly developed Web 2.0 portal with very little further development.
- **Multiple themes**: Several sets of UI representation, a.k.a. themes, could be developed for and adopted by one Web 2.0 application. The service providers could easily customize the user interface for their customers through creating a new theme or modifying an existing one. The end users are also allowed to switch the themes in various styles based on their preferences.

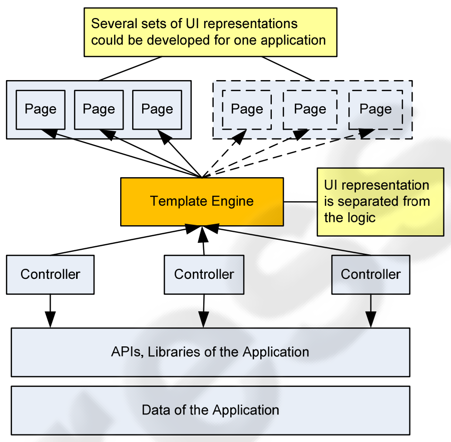

Several sets of UI representations could be developed for one application
Page
Page
Page
Page
Page
Page
Template Engine
UI representation is separated from the logic
Controller
Controller
Controller
APIs, Libraries of the Application
Data of the Application

Figure 8: Separation of UI representation.

Figure 9 demonstrates a typical workflow of serving a user request with template engine involved. The template engine, based on certain pre-defined algorithms and configured policies, automatically identifies the right page to generate the UI representation. Furthermore, the rendered page is cached to improve the efficiency of the UI layer by eliminating unnecessary rendering cost for the same page fed with exactly the same data.

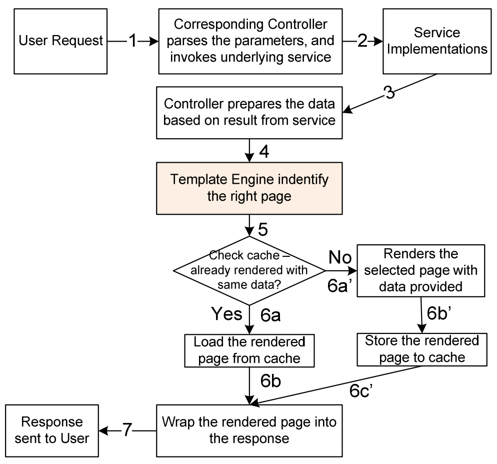

User Request
1
Corresponding Controller parses the parameters, and invokes underlying service
2
Service Implementations
3
Controller prepares the data based on result from service
4
Template Engine indentify the right page
5
Check cache - already rendered with same data?
No
6a'
Renders the selected page with data provided
6b'
Store the rendered page to cache
6c'
Yes
6a
Load the rendered page from cache
6b
Wrap the rendered page into the response
7
Response sent to User

Figure 9: Flow of serving a user request.

3.4 *SharK* External APIs

SharK external APIs provide a way for developers to interface with hosted *SharK* services or conduct customizations to construct *SharK*-based Web 2.0 portals.

Each API is based on REST protocol so that it can be invoked by sending HTTP GET or POST requests to the *SharK* server. The API specification defines language-independent protocols for invoking a service, thus can be implemented in any programming language.

In general, SharK external APIs consist of a set of interfaces for *authentication*, *blog*, *forum*, *wiki* etc. Each interface defines an interface name, a set of request parameters, and the DOCTYPE definition of the response package. For example, the interface for getting a list of photos from an Album has the name of "*shark.album.photo.list*" and the request parameters of "*vendorKey*", "*sessionKey*" and "*albumId*". This interface can be invoked by sending to the *SharK* server the following HTTP GET request: "*http://SHARK_SERVER/server/handler?method=shark.album.photo.list&vendorKey=THE_VENDOR_KEY&sessionKey=THE_SESSION_KEY&albumId=THE_ALBUM_ID*". The *url* might exceed the 255-char limitation of a HTTP GET, so it is encouraged to invoke the interface with HTTP POST. The response from an API innovation is an XML package, which is platform-independent.

4 REAL-LIFE DEPLOYMENTS

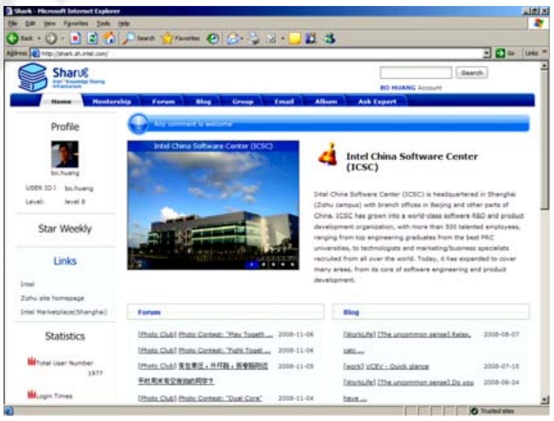
When developing *SharK*, we use many 3rd party tools to accelerate the development progress. Although different languages are used in implementing different modules, the application layer is developed in PHP using the *Symfony* framework. Actually some open source tools are modified to make them work well inside the *SharK* architecture. For example, we modified *Lucene* to make it work smoothly on top of the Unified Content Model, and added many codes to make *JXTA* fully integrated into *SharK*.

We built a reference Web 2.0 knowledge sharing portal based on *SharK* and customized it to three real-life Web 2.0 portals (Section 4.1-4.3). Although each of those three Web portals is currently deployed onto a single server, actually *SharK*-based Web 2.0 knowledge sharing portal can be flexibly deployed in multiple ways. Although deploying a *SharK*-based Web 2.0 portal to multiple machines and tuning its scalability on machine cluster belong to our future plan, actually a *SharK*-based Web 2.0 portal can scale very well even if all components are deployed onto one server. Our experimental result shows that single-machine-deployed *SharK* can scale well to serve for ~500 **concurrent** users.

Among three *SharK*-based real-life Web 2.0 portals, the 1st one is actually our reference portal, with which the construction of the 2nd one only took 2 weeks for one experienced software engineer and one experienced UI engineer. Since the 3rd portal has very different UI style, the customization took one experienced software engineer and one experienced UI engineer one month effort.

4.1 Enterprise Knowledge Sharing

In the spirit of eating our own dog food, we deployed an internal Web 2.0 portal for the purpose of knowledge sharing among employees inside an enterprise. As shown in Figure 10, communication among employees and cross organizations can be achieved through *Blog*, *Forum* and the *Web email* inside the portal. Sharing of big files is transparently supported by the underlying layer P2P services. *Mentorship* application is a new application for bridging mentorship among employees. The "*Ask Expert*" application is actually a customization based on *Forum*, which targeting for *Q/A* between technical experts and other employees. To bring better user experience, we also add the *RSS* support to allow users to subscribe forum articles and add the feature that allows employees to complete the article post to *Forum* through emails.



Intel China Software Center (ICSC)

Intel China Software Center (ICSC) is headquartered in Shanghai (Zizhu campus) with branch offices in Beijing and other parts of China. ICSC has grown into a world-class software R&D and product development organization, with more than 500 talented employees, ranging from top engineering graduates from the best PRC universities, to technologists and marketing/business specialists recruited from all over the world. Today, it has expanded to cover many areas, from its core of software engineering and product development.

Figure 10: Enterprise internal knowledge sharing portal.

4.2 *SchoolSpace*

The *SchoolSpace* portal works as a content aggregator for education mobile SMS (Short Message Services) and a backend social networking portal that links together parents, teachers, schools and students. When online, teachers, students and students' parents can have interaction through those Web 2.0 applications such as *Blog*, *Forum*, etc. Students' parents can even receive short messages sent to their mobile phones without logging onto this portal. Those messages include their kids' performance in school, school administration notifications, their kids' daily homework, recommended articles on this web portal etc.

4.3 Remote 1:1 Coaching

This portal serves for the purpose of bridging enterprise volunteers and students in rural areas to conduct remote 1:1 coaching. Students' profiles and enterprise volunteers' profiles are used for match-making to build 1:1 coaching relationship. The coaching is typically done through the *instant massager*, while each user can also use *Forum* or personal *Blog* for sharing mindset/knowledge/ coaching feedbacks. Internal email is used for asynchronous communication too.

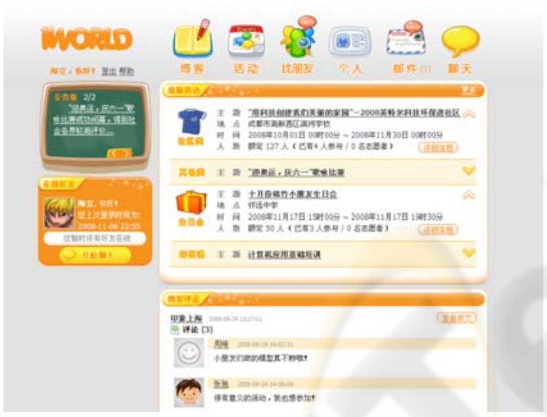


Figure 11: Portal to facilitate remote 1:1 coaching.

5 RELATED WORK

SharK presents a novel design of the Web 2.0 service infrastructure by integrating different Web 2.0 applications (e.g., *Blog*, *Wiki*, *Relation*, *Tagging*, *Ranking*, *Searching*, *IM*, *P2P* etc.) that enable effective and interactive knowledge sharing.

SharK has some similarities with Business suite 2.0 (an integrated software suite with typical Web 2.0 applications), and with many Internet forum applications (such as PHPWind, Discuz! and vBulletin). On the other hand, Business suite 2.0 is heavily adapted to mass collaborations using *Blog*, *Wiki* and *RSS* feeds, and those Internet Forum applications focus more on forum-style discussions using topic threads. They all lack some applications for knowledge sharing (e.g., *IM* and *P2P*), which are important features in *SharK*. In addition, different applications in Business suite 2.0 use different data model and rely on data adapters for the integration. In contrast, *SharK* supports different applications using Unified Content Model and standard content services for better integrations and better extensibility.

Finally there is a wealth of Web 2.0 applications in the Internet, such as *Facebook* and *MySpace* for social networking, *YouTube* and *Flickr* for user geneted content sharing. Though those applications have different emphasis than *SharK*, they apparently shares some common features and goals with SharK. Unfortunately, little details of those application designs have been published to date.

6 CONCLUSIONS

This paper presents several key design considerations of *SharK*, a Web 2.0 service infrastructure specifically designed for knowledge sharing. The adoption of the Unified Content Model and UI separation methodology lay out a solid foundation for *SharK*, which makes it a unique extensible platform for fast Web 2.0 knowledge sharing portal constructions. Three real-life *Shark*-based Web 2.0 portals clearly demonstrate the effectiveness and efficiency of *SharK*-based deployments. Although we are focusing on knowledge sharing in this paper, actually SharK can be easily customized to create other categories of Web 2.0 portals.

REFERENCES

Styles, C. (2006). How Web 2.0 will change history. From http://catherinestyles.wordpress.com/2006/08/27/how-web-20-will-change-history/.

Williams, J. B. & Goldberg, M. (2005). The evolution of e-learning. *Ascilite 2005.*

Drasil, P & Pitner, T. (2006). e-Learning 2.0: Methodology Technology and Solutions. *Proceedings of the International Conference ICT in Education. Vol. 1, 2006.*

Kesim, E. & Agaoglu E. (2007). A Paradigm Shift in Distance Education: Web 2.0 and Social Software. *Turkish Online Journal of Distance Education, Vol. 8, No. 3.*

Khalifa, M. and Lam, R. (2002). Web-Based Learning: Effects on Learning Process and Outcome. *IEEE Transactions on Education, Vol. 45, No. 4.*