D5.2v2
Integration of Plugins

Maintainer/Editor-in-chief	Aliaksandr Birukou
Core authors	Marcos Baez, Aliaksandr Birukou, Fabio Casati, Aalam Wassef
Maintainers/Editors	
Reviewers	Peep Kungas, Nardine Osman, Judith Simon
LiquidPub research group leaders	Fabio Casati, Roberto Casati, Marlon Dumas, Ralf Gerstner, Fausto Giunchiglia, Maurizio Marchese, Gloria Origgi, Alessandro Rossi, Carles Sierra, Yi-Cheng Zhang
LiquidPub project leader	Fabio Casati

Grant agreement no.	213360
Project acronym	LiquidPublication
Version	v2.0
Date	April 15, 2011
State	Solid
Distribution	Public

Disclaimer

This document is part of a research project funded by the European Community as project number 213360 acronym LiquidPublication under THEME 3: FP7-ICT-2007-C FET OPEN. The full list of participants is available at `http://project.liquidpub.org/partners-and-contributors/liquidpub-teams`.

Abstract

This deliverable describes how the services of the LiquidPub platform are integrated.
Keyword list: **integration**, **services**, **plugins**, **LiquidPub platform**

Contents

1 Architectural Overview and Rationale

As in all prototype deliverables, this document serves simply as a pointer to the software artifacts, along with a brief illustrations of what they represent.

This document is a prototype deliverable. However, before pointing to the software artifacts, we recap the LP architecture (to make this document self-contained and provide a proper reading guide) and also report on the status of the integration. We begin by describing the LP high-level architecture, **with text taken from D5.1v3** [3].

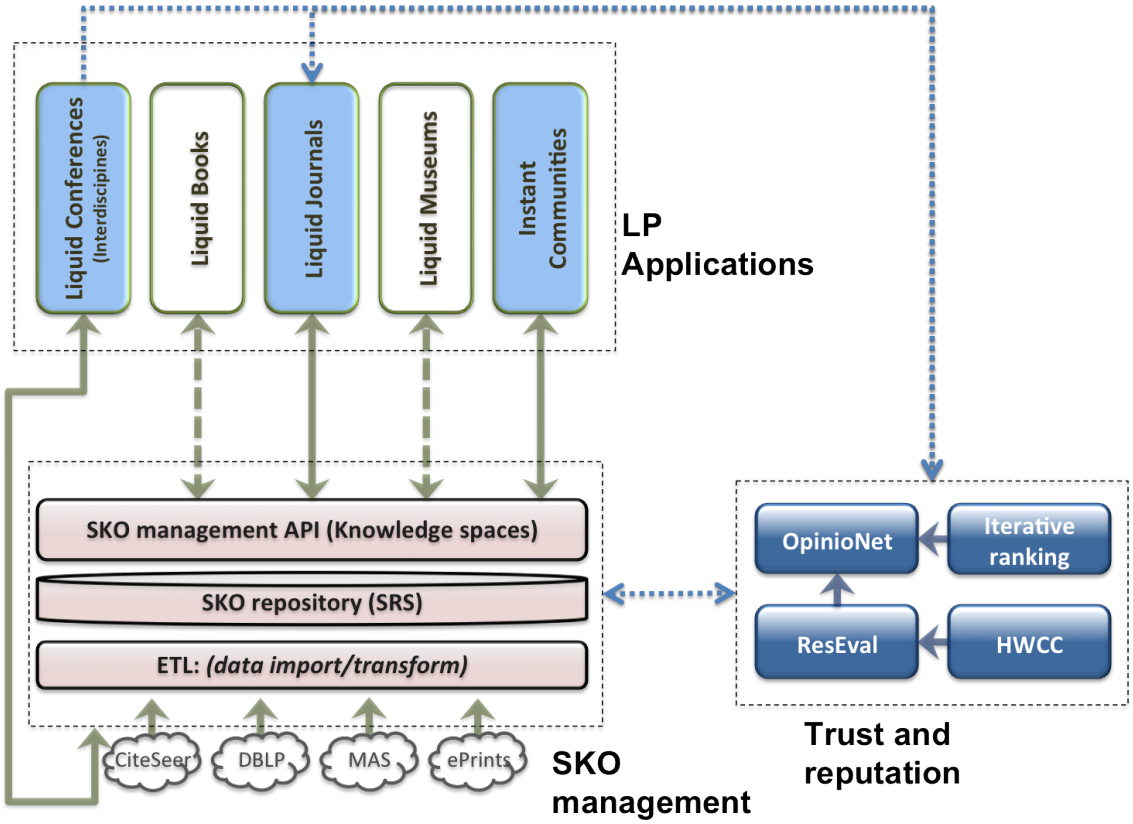

Figure 1: LiquidPub platform. Shaded rectangles represent implemented applications, while white rectangles show applications being implemented. Blue dotted arrows show how the flow of the reputation information, while the solid (or dashed, for non-implemented parts) lines show the flow of the information about the SKO. Finally, blue solid arrows in the trust and reputation module show the flow of the trust information

The goal of LiquidPub is to capture the lessons learned and opportunities provided by the Web and open source, agile software development to develop concepts, models, metrics, and tools for an efficient (for people), effective (for science), and sustainable (for publishers and the community) way of creating, disseminating, evaluating, and consuming scientific knowledge. This is realized through concepts and models which are supported by an IT solution (the LiquidPub platform) and an ecosystem of services. The way in which the LiquidPub platform (Figure 1) supports the LP

vision is the following:

- *LiquidPub applications* support different ways of capturing, sharing, and carrying on a *conversation* about knowledge. These applications provide metaphors and interaction patterns for the different contexts in which one can capture and disseminate knowledge. For example, *liquid conferences* facilitate conversations on papers (or parts of a paper), while community-oriented applications like *instant communities* aim at capturing implicit knowledge about scientific contributions and sharing it within a community of interest. Within the LP platform, the applications are represented as the vertical rectangles in Figure 1.

- A Scientific *Resource Space Management System* (SRS) stores and manages SKOs and allows access to them. It includes

  - A SKO repository, that models information collected by the different applications, and also acts as an integrated repository of liquid knowledge.
  - A SKO management API, called Knowledge Spaces, that allows both to access SKOs as well as ways to group them and define access rights.
  - Extract Transform Load (ETL) routines and wrappers that connect LP with existing sources of information and in general with the non-LP world. An example of this consists in getting metadata about scientific publications from external sources such as CiteSeer, DBLP, Microsoft Academic Search (MAS) and ePrints via API or wrappers to the LP platform so that such information can be integrated into the SKO repository or the applications, or can be used by the reputation tools.

- A *trust and reputation* module that processes information from the SKO repository and from the applications to compute metrics that (i) reflect the collective opinions of scientists over scientific contributions, (ii) go beyond metrics with well-known flaws such as citation counts, and (iii) expose metrics that encourage behaviors that are beneficial for science. This module includes OpinioNet, Reseval, Iterative ranking, and Homophily Weighted Citation Count (HWCC).

While advancing with the design and implementation of the applications, we progressively understood that in terms of the implementation strategy, the following aspects have to be considered:

1. The LiquidPub applications and other parts of the LiquidPub architecture that ultimately deliver the LP vision to the user *evolve over time*, as we have already seen during the life of the project. One of the main reason for the evolution of the applications is that only once they become adopted we start learning how people use them. We feel this evolution is natural as the concepts are novel. As we dive deeper into the development and as we have early prototypes available, we better learn and understand requirements as well as how to refine and extend the basic abstractions and underlying conceptual models. We cannot imagine getting the things right from the very beginning. For instance, Instant Communities contains many features that we prototyped in Liquid Journals and envisioned in LiquidBooks. We did not even foresee such tools in the beginning of the project.

2. There is no unique way to capture and share knowledge. We do it differently in different domains. Consequently, there is no single interaction model or metaphor that can fit all needs. However, there are commonalities that we can exploit in the ways we model knowledge, knowledge sharing, and knowledge evaluation-related actions.

3. Researchers in the area of Science 2.0 may develop other dissemination paradigms, or paradigms targeted to specific artifacts (or, we may find over time that many concepts from books, conferences and journals will merge - as we indeed already started to experience with Liquid Museums and Instant Communities). For example, some researchers outside LiquidPub (from UNSW Sydney) developed a *Liquid Benchmark* module, while others are interested in using Liquid Journals as a way to annotate, tag, and link resources in a digital library *they* own, for example a library of *demos* (as in the Share [1] system from TU Eindhoven). These two modules developed by other researchers can already interface with LiquidPub.

These observations carry two important implications. The first implication is that it would be a mistake to make the abstractions of each of the applications tightly coupled. In fact, both from a conceptual perspective and in terms of implementation, in LP we proceeded by developing the separate dissemination paradigms for each application. Each paradigm and supporting tool (including in particular the UI and the design of the interaction with the user) has its own requirements and abstractions. It is important that, at least initially, the abstractions and the conceptual model are tailored for each of them, because this makes it easier to provide a coherent and understandable set of concepts at the appropriate abstraction level. It also makes easier to write code and design the user interaction. For example, Liquid Conferences have the notions of *conference*, of *paper* to be discussed, of *comments*, etc. Liquid Journals have the notion of *scientific resources* that are the items in a *journal*, the notion of journal itself, the notion of *issues*, *editors*, etc. Liquid Books have *books* and *editions* as first class objects. Instant communities have concepts such as *panels*, *panelists*, *presentations*, or *questions*.

Furthermore, as the tools are developed and used, and even as our thought process proceeds and new ideas and opportunities come to mind, these conceptual models and, more in general, the functionality offered by each of the applications evolves. Because of the research nature of the project, binding the models and implementations tightly from the start would reduce the flexibility and make it more difficult to capture the lessons learned during development and early usage.

However, we *do* want to have an integrated end result because we do want each LP application to benefit from what the others can provide. While we believe that a certain degree of autonomy is necessary as it facilitates flexibility and evolution of ideas (and tools), it is also important to connect the applications so that they can share knowledge and put the functionality of one at the service of the other. For example, a paper in a liquid conference can become a scientific resource in a liquid journal or can be shared with a community.

The second implication is that we need to provide the hooks for other researchers to build on top of our results and to be able to easily integrate their idea with LiquidPub ideas and tools. We envision different kinds of extensions, some will be directly extending the applications, but others can build directly on top of the SRS and Knowledge Spaces, as discussed below. This is what we mean by the "ecosystem": providing a way for other R&D efforts to integrate with LiquidPub

[1] `http://is.tm.tue.nl/staff/pvgorp/share/`

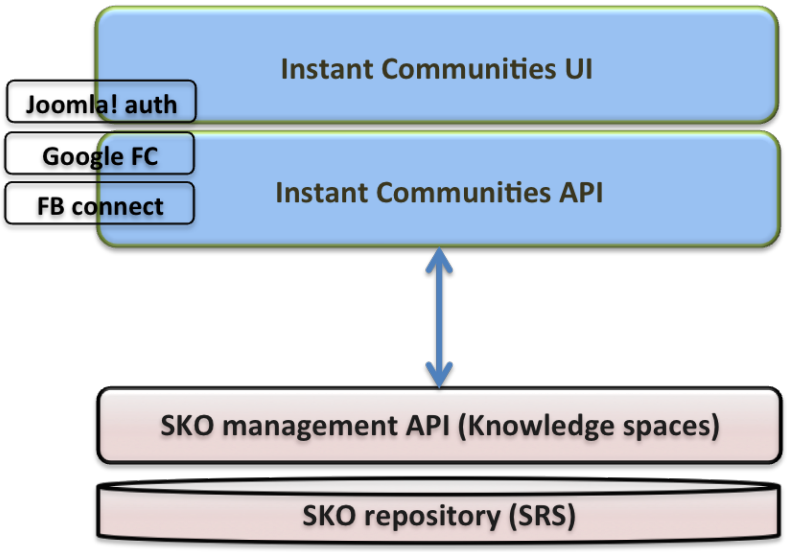

Figure 2: Instant Communities architecture

(such as Liquid Museums, in development with the Cambridge Museum of Archeology), leveraging what we can offer in terms of liquid knowledge but also providing information that can be collected as SKOs, measured by the LiquidPub reputation metrics, and provided as information to the applications.

These problems (integration and extensibility) are the LiquidPub counterpart of the well-known *enterprise application integration* (EAI) research and development area, where the goal is to integrate a company's IT systems and services. EAI is needed in an enterprise because systems are developed independently (and it is important that it is so) but then they also need to be integrated, otherwise it is impossible to execute business processes efficiently as these invariably span many IT systems. The response to this was the development of a multi-billion dollar industry around such concepts as enterprise middleware, service bus, and the like. LiquidPub needs the same kind of bus, but for knowledge and for supporting knowledge dissemination. This knowledge bus is represented by the SKO management module, where abstractions related to scientific knowledge are integrated and can be transferred across applications. Having a knowledge bus also implies having a model for liquid knowledge at a lower level of abstraction and more general in that it needs to cover the concepts of the applications (and of the ones that will come along in the future, to the possible extent). So the challenge in the knowledge bus and its model is to be generic enough to enable integration and extension but also specific enough to facilitate the development of services for the LiquidPub ecosystem.

Besides being a bus that allows interactions among applications, the SKO management module includes a repository for those applications that do not maintain one and wish to reuse the central repository, like instant communities and liquid journals. These applications have their own UI and even the supporting API that has application-specific concepts and metaphors. This API then invokes the (application-independent) knowledge spaces API thereby mapping specific concepts into SKOs and SKO management operations. An example of the architecture of the Instant

Communities application is given in Figure 2.

The trust and reputation module has the same underlying principles: some aspects can be made generic and as such insist on the the SKO repository and SRS. Other parts are inherently application-specific and integrate with the applications (also because they have grown and were elaborated with the application and depending on the information that the application provides). For example, the OpinioNet reputation module was developed for the general case of reputation handling and then separately integrated with Liquid Conferences and Liquid Journals applications. OpinioNet takes information from the Iterative Ranking algorithms and from the applications and provides reputation information back to the applications directly, while Reseval exploits the integration of SKO metadata at the knowledge bus level. This means that reputation in LP entails mapping reputation-relevant information not only from the applications, but also from the external sources such as DBLP and Microsoft Academic Search into the shared model and then operating on this to derive reputation metrics for people and scientific artifacts.

What this all means in terms of architecture and in terms of LP development process (development of both concepts and software) is the following:

- The applications initially had their own abstraction and conceptual model. These were also the initial requirements for a knowledge bus/shared model.
- Over time, some of the applications converge towards the common model as represented in the SRS, at varying degrees of integration. For example liquid journals and instant communities still have an own UI and a thin API layer, but now insist on top of the same Knowledge Space API and SRS modules for SKO management. Liquid conferences are also integrated via an adapter though they also maintain their own repository.
- Over time the shared model evolves to accommodate new requirements at the appropriate level of abstraction. In other words, in some cases new concepts in the application may simply correspond to new mappings, while in other cases may result in new abstractions to be pushed down to the level of the shared model to SRS or Knowledge Spaces level.

This strong emphasis on *applications* (which we see not only as use cases, but rather as key elements of how LiquidPub is exposed to users) was not part of the initial description of work, and so the applications are not described in other prototype deliverables. Therefore, we include here pointers to the prototypes (for journal, conferences, and communities only, since books and museums are not implemented yet, though we have detailed contract models and requirements). The other software artifacts are discussed in the other prototype deliverables, namely: D1.3v2 [1] for SRS, D4.3v2 [2] for the reputation tools, D5.1v3 [3] for a more detailed description of the architecture. Liquid Journals, Liquid Conferences, Instant Communities applications and Knowledge Spaces module are described in further sections of this deliverable. We also observe that the evolution of our thinking in terms of architecture means that what we have is not really a main platform with plugins to be added (the initial thinking), but rather full-fledged software applications to be integrated, many of which even have their own UI to interact with the end users.

We now report on the status of the integration among the different components of the LiquidPub architecture. As we will see, in this version 2 of the deliverable we have achieved a better level of integration since the components became more stable and understood. The current integration status is as follows:

- the Liquid Conference platform is integrated with the SRS via an adapter that ports LC content into the SRS and makes it therefore available to all modules that plug to the SRS
- the Instant Communities application is integrated with the SRS and Knowledge Spaces
- the Liquid Books platform is conceptually integrated with the SRS and Knowledge Spaces, meaning that currently we have a specification of the behavior of the adapter between books and these tools, but not yet an implementation.
- Knowledge Spaces are integrated with SRS meaning that it provides personalization layer on top of SKOs in SRS.
- Reseval is integrated with SRS, meaning that it uses SRS to access information about SKOs and researchers for computing citation statistics.
- Reseval is integrated with the Homophily Weighted Citation Count (HWCC) component and exposes HWCC metrics in the Reseval API.
- OpinioNet uses Reseval to gather basic citation data. What this mean is that currently we can compute reputation metrics based on usages of knowledge artifacts in the Liquid Journal platform.
- OpinioNet uses Iterative ranking algorithms to help compute the reputation of reviewers based on their past review performance (this is ongoing work)
- OpinioNet reads Interdisciplines database to help compute reputation measures for the Liquid Conferences application, as it also reads Liquid Journal data to help compute reputation measures for the Liquid Journals application (we note that in the case of Liquid Journals, reputation measures are sent back to the Liquid Journal UI for display)
- Microsoft Academic Search and DBLP, to provide a bootstrap for existing knowledge, are integrated with SRS via ETL

2 Liquid Journals

2.1 Source Code

Liquid journal is released as an open source project under the GNU Affero General Public License (AGPLv3) [2]. At the moment, the source code is available at `https://dev.liquidpub.org/svn/liquidpub/prototype/liquidjournals/` (using lp-guest and lp-password credentials).

The code is organized in two separate sets of components: the service side which implements the liquid journal features and exposes them as RESTful services, and a set of client applications (web, mobile and browser plugins) that offers users the interface for interacting with liquid journals.

[2] `http://www.gnu.org/licenses/agpl-3.0.html`

2.2 Current status

We have implemented the services for creating, filling and browsing liquid journals as well as the features for sharing, searching, annotating, linking and navigating scientific resources. All these services are provided by a backend application available at `http://liquidjournal.org/api/`. On top of these services we have developed:

- a first prototype of the Web UI that implements the user interface for the features provided by the backend,
- an iPhone application that puts liquid journals into the most widely used phone in the market,
- an email interface that allows us to push all the data that is usually shared with colleagues by email, into the liquid journal platform, and
- a browser plugin to collect scientific resources and to fill liquid journals while browsing the web.

2.3 Videos of demos

You can find videos of how Liquid Journals work at `http://liquidjournal.org/demo.html` along with links to some live examples.

2.4 Architecture and API

The information for developers is available at `https://launchpad.net/liquidjournal`, where we provide links to the project wiki, homepage and other resources. In particular, the API specification is available at `http://liquidjournal.org/api/`.

A detailed description of the architecture of the Liquid Journals can be found in [3].

2.5 Other documents

More information about Liquid Journals, such as presentations and related documents, is available at `http://project.liquidpub.org/research-areas/liquid-journal`.

3 Liquid Conferences/Interdisciplines

Interdisciplines is an online platform allowing the management of text-based conferences. It implements the Liquid Conferences use case and is available at `http://www.interdisciplines.org`.

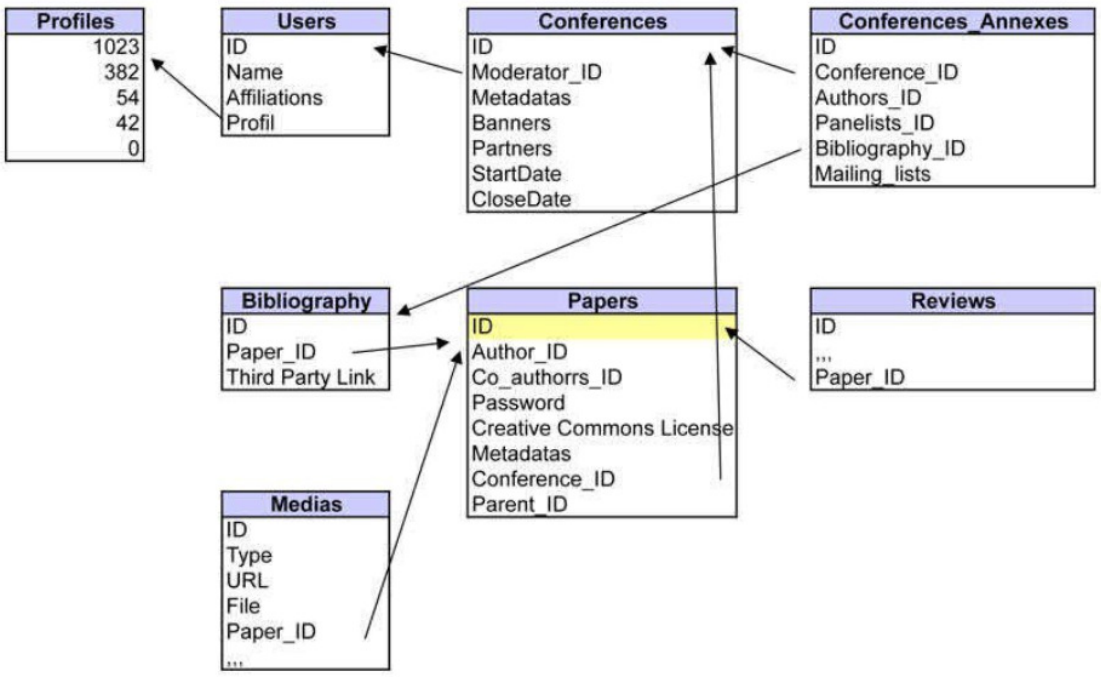

Figure 3: Database of Interdisciplines

3.1 Source Code

The source code of Interdisciplines might be released later.

3.2 Current status

Interdisciplines now supports main features of the Liquid Conferences use case: modular paper structure with authors specified for each section, managing bibliographies, managing paper licenses, reviewing and commenting papers. It has been already used for organizing a conference on Governing the Future (summer 2010) and the LiquidPub workshop on Trust and Reputation (autumn 2010 - spring 2011).

Future extensions of the platform include login with Academia.edu account and managing paper bibliographies via Mendeley and sharing them.

3.3 Videos of demos

Deliverable D5.1v3 [3] contains screenshots of the Interdisciplines platform.

3.4 Architecture

Interdisciplines' architecture is composed of a backend and a frontend (public website). The backend relies on a relational database whose simplified overview can be seen in Figure 3. The database includes the concepts of users, profiles, conferences, additional information about conferences as represented by annexes. It also contains the information about papers, and different artifacts related to the papers: bibliographies, media (such as video), reviews.

The presentation at `http://www.slideshare.net/ColDev/interdisciplines-20` provides a brief overview of Interdisciplines.

4 Instant Communities

4.1 Source Code

Instant Communities has been also made available to the open source community to help us improve and maintain the application. The open release includes the IC backend service and the web client, both under the AGPLv3 licensing scheme. Information about how to contribute can be found at `http://base.kspaces.net/confluence/display/IC/Instant+communities` under *development guidelines*.

4.2 Current status

The Instant Communities tool is currently in alpha stage. The latest snapshot can be found at `http://alfa.instantcommunities.net`, with new updates being deployed every two weeks (the regular sprint duration) according to the tool roadmap.

The tool will be providing support for seminars, panels and conference sessions during the summer 2011 in several universities and at several conferences and workshops. These pilots will help us to validate not only the concepts but also the effectiveness and usability of the entire platform.

4.3 Videos of demos

You can find a video of how Instant Communities work at `http://open.instantcommunities.net/` along with links to some live examples. The test version of Instant Communities is available on line at `http://test.kspaces.net/ic/#/events`. It is not yet open for general usage, so please contact us if you are interested.

4.4 Architecture and API

You can access the Wiki of the project at `http://base.kspaces.net/display/IC/Instant+communities` to get the latest updates on the architecture and API. As the project evolves to a more mature product, we will be adding more end-user oriented documentation. A detailed description of the architecture of the Instant Communities can be found in [3].

5 Knowledge Spaces

5.1 Source Code

Knowledge Spaces (KS) is at the bottom of the LP infrastructure, providing services to other LP tools such as Instant Communities. Thus, to allow a more dynamic evolution and response to the changing requirements of the scenarios covered, we release KS as open source. The project is released under the AGPLv3 licensing scheme. Information about how to contribute can be found at `http://services.kspaces.net` under *development guidelines*.

5.2 Current status

The module is currently under development with the core functionality already implemented. The latest release is in alpha stage and available only to our infrastructure but we plan to open it as hosted service in the future.

5.3 Architecture and API

We refer the reader to the latest version of the architecture and API documentation at the developers page: `http://services.kspaces.net`. Further details of the architecture can be found in [3].

References

[1] D1.3v2 liquidpub core platform (prototype), 2011.

[2] D4.3v2 development of the analysis and evaluation plug-ins (prototype), 2011.

[3] D5.1v3 design of the liquid publications integrated platform, 2011.