Erasmus Without Paper
from the technical perspective

Janina Mincer-Daszkiewicz
Wojciech Rygielski
University of Warsaw
Agenda

- Project goals
- Development decisions
- Architecture
- Security
- Use cases
- API + flowcharts
- Summary
Erasmus Without Paper (EWP)

• 11 partners from public institutions, higher education organizations, and companies from 8 European countries.
• 11 associate partners.
• Dissemination potential to over 400 HEIs from 36 European countries.
Project goals

• Design and work out a pilot for an integrated communication network supporting the exchange of student data\(^1\) in an electronic form.

• Build connecting software modules that will allow Student Information Systems (SISs) with built-in mobility modules\(^2\) and/or stand-alone Mobility systems to exchange data over the EWP Network.

\(^1\)data, not documents (eg. scanned copies), which can be processed automatically, stored in databases, used to create documents

\(^2\)part of SIS that takes care of Bilateral Agreements, student applications, Learning Agreements, Transcript of Records and other documents
Project goals – details

• Describe mobility scenarios.
• Create common data models for the exchanged information and design appropriate document formats.
• Define transport protocols and standards.
• Take care of identity management (authentication and authorization methods).
• Solve the security and privacy issues.
• Build connector modules that allow data handling software to send and receive data over the network.
• Include extra tools to increase performance and usability (e.g. grade conversion).
Project development decisions

- Open source approach.
- General overview of documents and specifications on [developers.erasmuswithoutpaper.eu](developers.erasmuswithoutpaper.eu).
- Design and implementation reported on GitHub.
- Specifications available publicly, everyone can contribute (not only EWP partners).
- Changes easy to follow (version numbers, release notes).
- Set of repositories for various sections of documentation and code.
- Backward compatibility.
- API and data formats defined formally by XSD.
- Tool for formal verification of the produced data files.
Approving Learning Agreements

At some point, the student wants his LA to be approved. To do so, he first approves it himself, then waits for other actors to approve it. As with the editing, the approval process can also be started by other actors (e.g., the receiving coordinator). Regardless of who starts the process, the LA is approved when three "approved" entries are recorded in a row. At this moment, LA gets "approved by all parties".

Learning Agreements can still be edited after they are approved. Then, they can be approved again. Each such change is recorded, and all actors can review each of these changes.

There are no new APIs needed for approving LAs. We will be using only the ones we have described earlier. The following flowchart presents the entire process.

---

### Approving Learning Agreements

<table>
<thead>
<tr>
<th>Student being sent</th>
<th>Sending Coordinator</th>
<th>Sending Web App</th>
<th>Receiving Web App</th>
<th>Receiving Coordinator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wants other actors to approve current content of the LA</td>
<td>Gets notified about</td>
<td>&quot;Approved by the student&quot; history entry is appended.</td>
<td>Outgoing Mobility CNR API</td>
<td>Might get notified about a new LA version being approved by the</td>
</tr>
</tbody>
</table>

Note: This process can be initiated by any actor.
EWP Network

- EWP Network is composed of EWP Hosts and Registry.
- Each EWP Host may represent more than one HEI.
- EWP Host publishes Discovery Manifest File, somewhere on its servers. The manifest is fetched by the registry, information is extracted and propagated.
Registry

• The only central part of the EWP Network.
• Keeps track of the EWP Hosts and APIs they implement (possibly only subset).
• Updated automatically – periodically reads Discovery Manifest files (which are updated locally → scalability).
• API – Discovery (obligatory), Echo (for security testing), Event listener, other.

```xml
<?xml version="1.0"?>
  <ewp:dev-email>admin-or-developer@example.com</ewp:dev-email>
  <ewp:dev-notes>It's not a real manifest. Just an example.</ewp:dev-notes>
</manifest>
```
Two types of certificates involved in the communication:

- **Server** certificates
  - used by the Host when it **responds** to API requests,
  - "regular" SSL certificates, bound to host’s domain, signed by a trusted CA,
  - neither the clients nor the registry will be storing server certificates.

- **Client** certificates
  - used to **issue requests** within the EWP Network,
  - each Host (via its Manifest file) declares a list of certificates it will use for making requests to other hosts, list is fetched by registry, **fingerprints** of these certificates are served to the EWP Network,
  - **Extended Validation (EV)** certificates are recommended (but not required) for serving manifest files, because they allow the Registry Service administrators to vet new EWP partners more easily. They are DNS-spoofing-proof and Terena provides them for all HEIs **for free**.

Use cases

• Based on a survey – 1049 filled questionaries from 31 countries.

• Use cases identified
  – Interinstitutional Agreement
  – Nominations
  – Learning Agreement
  – Arrival & Departure
  – Transcript of Records
  – Grade Conversion

• Summary
  – Very high interest in EWP
  – All steps of mobility are strong candidates for EWP integration
  – IT platforms less used for data exchange than ... snail mail
  – Local IT systems only modestly integrated

• Detailed analysis of use cases led to design of API
The goal of Erasmus Without Paper is to enable different IT systems to seamlessly exchange information and documents among Erasmus partner institutions. How do you think achieving this goal could impact your work?

Answered: 903  Skipped: 0

- Positively: 89.3%
- Negatively: 2.7%
- It would not have a...: 8.1%

Is the data of outgoing Erasmus+ students for the following documents/steps stored in a local system in such a way that it can be reused for other purposes (e.g. reporting to the Mobility Tool+ or used for local documents like the diploma supplement)?

Answered: 903  Skipped: 0

- Inter-institutional agreements:
  - Yes: 54%
  - No: 32%
  - No, but we are considering to implement a tool/solution: 11%
  - I don't know: 4%

- Erasmus+ nominations:
  - Yes: 48%
  - No: 36%
  - No, but we are considering to implement a tool/solution: 11%
  - I don't know: 5%

- Learning Agreement:
  - Yes: 49%
  - No: 33%
  - No, but we are considering to implement a tool/solution: 16%
  - I don't know: 2%

- Confirmation of student’s...:
  - Yes: 48%
  - No: 36%
  - No, but we are considering to implement a tool/solution: 14%
  - I don't know: 3%

- Transcript of records (from...):
  - Yes: 47%
  - No: 34%
  - No, but we are considering to implement a tool/solution: 14%
  - I don't know: 5%
Use case leading to API

Learning Agreement

Institution 1
- Learning Agreement proposal
  - N
  - Accept?
  - Y
  - Learning Agreement

Institution 2
- Learning Agreement proposal
  - Accept?
  - Y
  - Learning Agreement

Editing a Learning Agreement (LA)

- Student being sent
  - Nomination has been accepted
  - Student is notified that he should now fill his LA.
  - Wants to add new courses to his LA
  - There are couple of ways LA can be edited, depending on the course-related APIs implemented by the receiving institution.
  - Courses API implemented?
    - No
    - Yes
      - Courses Search API implemented?
        - Yes
          - Courses are saved in the LA
        - No
          - Sending Web App can try to use Course Search API to provide the student with a simple search engine.
  - Note that sending coordinators and receiving coordinators are also allowed to edit students' LAs. This flowchart presents only one of the possible use cases (the one we find most common).
Accessing information on Institutions APIs

- Allow members of EWP Network to discover basic information on other institutions and departments covered by the network (**fact sheets**).
- **Institutions API** – e.g. address, contact persons, logo image, list of departments, list of academic terms used. May allow clients to fetch **PDF Fact Sheets** (nice, printable format, exchanged by IROs).
- **Departments API** – detailed information on specific departments, e.g. address, contact persons, institutes or other kinds of subunits.

### Institution and Department-related "Fact Sheet" APIs (examples of most basic use cases)

<table>
<thead>
<tr>
<th>Student or IRO staff (any HEI)</th>
<th>Local Web App</th>
<th>Foreign Web App (might equal local)</th>
<th>Other apps (e.g. IIA Repository)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wants to obtain contact details of foreign IRO</td>
<td>&quot;Proxy-like&quot; user interface</td>
<td>Institutions API</td>
<td>Wants to obtain email addresses of IRO staff members</td>
</tr>
<tr>
<td>Wants to obtain a link to online ECTS Catalogue</td>
<td></td>
<td></td>
<td>Wants to obtain IDs and names of all departments (e.g. for UI features like autocomplete)</td>
</tr>
<tr>
<td>Wants to obtain contact details of foreign faculty/department</td>
<td>&quot;Proxy-like&quot; user interface</td>
<td>Departments API</td>
<td>Wants to obtain more details on some departments</td>
</tr>
<tr>
<td>Wants to retrieve a &quot;legacy&quot; PDF document to read or print</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Change Notification Receiver (CNR) and Notification Senders API

- CNR is a callback URL for push notifications.
- Partners subscribe for notifications by implementing a chosen CNR API and publishing it in their manifest file.
- CNR URL is triggered whenever a related entity is updated. This allows the partners to keep fresh copies of data.
- Server responsible for the entity must be able to send such notifications (this ability is also published in the manifest files).
Interinstitutional Agreements (IIA) API

• Starting point of each mobility process.
• IIAs might be stored in a central EWP IIA repository.
• It might be a web application (with a user interface), which keeps track of all changes and provides the latest copy of the agreement to all partners.
• Final scenario to be supported still under discussion among partners (centralized vs decentralized approach)
Summary

• State of work in June 2016
  – Use cases and recommendations for developers.
  – Design of architecture, security, data flow.
  – First versions of API specifications (under review by the project partners).
  – Data model and data format – work has started.

• Soon
  – Design of data formats (data types for API parameters).
  – Registry available for testing.
  – Libraries for connectors.
Summary

• Work in progress, but EWP partners are open for discussion.
• Call for cooperation.
• Acknowledgments – tribute to all project partners for their tremendous job which makes it all happen.
Additional information

- EWP website: www.erasmuswithoutpaper.eu
- GitHub: github.com/erasmus-without-paper
- EWP for developers: developers.erasmuswithoutpaper.eu

Register for EWP Newsletter to keep in touch!