NM3: ¡strike¿web 2.0 mashups¡/strike¿

6700030 / project-based course, semester 1, 3 ects

Æliens

course description – nm3: web 2.0 mashups

The course description(s) are taken from the accreditation report Creative Technology (version 2.0).

contents The course presents advanced web technology, that allows for the development of data-driven dynamic web applications, using web services, such as google maps and AJAX, XML and JSON, in the Rich Internet Applications, provided by flex /as3.
Recommended literature: Professional Web 2.0 Programming, by Eric van der Vlist, Danny Ayers, Erik Bruchez, Joe Fawcett, Alessandro Vernet
Online reference(s):

- code.google.com
- www.adobe.com/devnet/flex

prerequisite(s): CS1 – computer & network architecture(s)

goal(s) & attainment target(s)
The course aims at providing

- awareness of available web 2.0 services, tools, APIs and SDKs
- familiarity the design and development life-cycle of data-driven rich-media applications
- fluency with flex in combination with AJAX and web services
- full literacy with developing moderately complex media rich applications

Students are expected to have a sufficient degree of creativity, and will be stimulated to explore the wealth of available technologies to make stunning applications.

place in curriculum: NM3 is an advanced course for NM students. As a follow-up on NM1 and NM2 it allows, in combination with CS3: Data Driven Applications, to build fully-functional, professional web sites, such as social community sites.

application area & motivating example(s) Current Web 2.0 sites allow for user contributed content, including media content such as images and video. many of the existing sites, such as youtube and flickr, moreover, offer APIs to use and store content by means of web services. Mashups allow for quickly developing prototypes, incorporating web services, that in it self may act as media portals, providing web services to for other communities. Many interesting sites of this type are developed in the domain of cultural heritage, in particular historical musems and museum of contemporary art.

teaching method(s) The course will be organised around lectures, which will introduce basic examples and which will provide an in-depth explanation of the technologies. The assignments will consist of a series of basic exercises and a final exercise In which the students are required to develop a moderately complex dynamic web application.

Regular feedback will be given in classroom sessions where students present their work as well as via online comments or email. Grading will be based on basic assignments, the final assignment project with documentation, as well as an essay in which a topic of choice, either technical or in relation to the application of web services in the development of digital learning or educational games, is discussed in more depth.

special facilities; computer lab & presentation facilities

course outline(s) – nm3: web 2.0 mashups

In this part a more detailed discussion will be provided of **topics**, **learning goals**, **materials** used, and the actual **structure of the course**, as well as a sketch of the **assignments** given. Also **references** to releveant literature is provided, including **online resources**. At the end, **advice for students** following the course will be given, as well as **hints for the instructor(s)**.

course topic(s)

Although the notion of **mashup(s)** suggests a quick configuration of services, there are many issues that must be taken into account when creating **serious mashup(s)**.

- data-driven applications – XML vs JSON
- client-side processing – AJAX vs Flex
- web-services – REST vs SOAP
- map(s) – geo-tagged data mashup(s)
- social network(s) – user contributed content
- experience design – immersive interface(s)
- security & privacy – exploit(s) & hack(s)

Each of the topics mentioned might require a course on its own. However the goal of the *web 2.0 mashup(s)* course, is explicitly to let students gain experience in creating mashups, and not to be intimately familiar with the underlying theory and technical issues.

learning terget(s)

Basic skills involve the use of technology, involving both programming issues, as well as service APIs.

- skill(s) – scripting, configuration, use of service APIs
- knowledge – APIs, protocols, REST & SOAP
- theory – web 2.0, social networks
- experience(s) – construction of moderately complex mashup
- attitude – craftmanship, creativity

However, not only technical issues are important, but also issues of **design** and **creativty** in developing novel **(combinations) of services**, together with an **appealing interface**.

lesson material(s)

Most of the material(s) that are currently available are related to the **flex/as3** technologies that we used for our **ximpel** interactive video platform.

- canonical example(s) – *flex* / ximpel
- (online) reference material(s) – adobe live docs / flex
- challenging target(s) – labs.adobe.com

Apart from the technical material, we also wish to refer to a number of **inspirational application(s)**, which are developed usinga variety of technologies.

inspiration(s) – (serious) mashups

- we feel fine – www.wefeelfine.org
- universe – universe.daylife.com
- twitter – twittervision.com
- flickr – flickrvision.com
- newsbreaker – www.newsbreakergame.com
- collage – www.francisshanahan.com/collage/bezos/bezos_b.aspx

These inspirations can be taken by the students as a starting point for developing the mashups, prereably in a **game context**, using their **technology of choice**.

course structure

The course does require active participation of the student(s), not only in exploring the technolgy by making the assignments, but also by presenting **solutions and problems** in class.

session(s)

1. introduction of mashup technologies
2. basic assignment(s) – map(s), flow(s) & diagram(s)
3. server technologies – the (W)AMP stack
4. technical issues – services & protocols
5. student presentation of concept final assignment(s)
6. services – flickr, amazon, google, etc.
7. interaction – model(s) of immersion
8. presentation of final assignment(s)

The course will take a **technology-agnostic** approach, favoring concept(s) over implementation(s) or language(s). Neverteheless, most of the examples presented are based on flex/as3.

assignment(s)

There will a small number of assignemts, to be made by the students individually. The goal of these assignments is to provide a structure that assists the students in exploring the technology. Basic assignemnts (may) include:

basic(s) – *web 2.0 mashup(s)*

1. flow(s) – presenting time-ordered data
2. map(s) – presenting geo-tagged information
3. diagram(s) – presenting abstract structure(s)

For the final assignment(s) of the course, students are allowed to work indidually, or in groups of two or three (maximally) students. Work done in groups must be proportionally more challenging and complex. Students can make a choice out of (among possibly others):

final(s) – *web 2.0 mashup(s)*

- map-based information system – using umap with flex/as3
- web-service driven game – e.g. based on RSS-feeds
- information game – e.q. collaborative fitness game

In effect, students will be encouraged to follow their own ideas, in for example implementing a game using visualization technology, giving information and (game-play) feedback in visually compelling ways.

reference(s)

There are a number of references on mashups for particular **API**s, of which a selection is given here. In addition, the student may need more detailed knowledge of how data manupulation and components are dealt with in **flex/as3**.

1. Flickr Mashups, by David A. Wilkinson, Wrox
2. Amazon.com Mashups, byFrancis Shanahan
3. del.icio.us Mashups, by Brett O'Connor
4. Web APIs with PHP, by Paul Reinheimer, Wrox
5. Professional Adobe Flex 2 (Programmer to Programmer) by Rich Tretola, Simon Barber, and Renaun Erickson
6. Visualizing the Semantic Web: XML-based Internet and Information Visualization by Vladimir Geroimenko
7. A. Eliëns, topical media & game development – media.eliens.net

A wealth of material and references can be found at my **topical media & game development** site, including tutorials and examples.

online resource(s)
A wealth of technologies is available. Of particular interest, apart from the **ready-to-apply (W)AMP stacks**, are tutorials on how to use **XML in e4x** scripting extensions, and the **mashup servers** of **wso2**. A new contendet, at the time of writing, on the RIA market is **JavaFX**, which, covering a wide variety of platforms, including mobile devices, is, given the wide-spread adoption of java, certainly of interest.

- **flex** – www.adobe.com/products/flex
- **xml** – tutorial / example(s)
- **umap** – www.afcomponents.com/tutorials/umap_as3 / flex tutorial
- **JavaFX** – www.javafx.com / www.sun.com/software/javafx
- web orb – www.themidnightcoders.com/weborb
- labs – labs.adobe.com/technologies/flex
- **api(s)** – labs.adobe.com/wiki/index.php/ActionScript_3:resources:apis:libraries
- code – code.google.com
- **db** – **sqlite** / mysql / exist/xml
- wamp – **appserver** / server2go / webdeveloper
- **mashup(s)** – **wso2**: wso2.org/projects/mashup
- amfphp – sourceforge.net/projects/amfphp
- **multiuser** – **smartfoxserver.com** / code.google.com/p/gfs-server / osflash.org/red5

Web service APIs are available in many language environments, including **flex/as3**, developed by **adobe labs**.

advice for the student(s)

Given the overwhelming number of web services, APIs, language environments and server technologies, you really have no other option than make a **choice** and **focus** on one particular API and try to get the most out of it. Do not forget the **issue of design**, and try to develop an **interface** that is both **functional** and **visually appealing**. An obvious choice is to take a **(google) map** component and combine this with one or more services. As an extra challenge, think of how to incorporate your mashup in a **game context**, or even make a **service-driven game**.

hint(s) for the instructor(s)

For a 3 credits course, obviously, the number of topics far exceeds the available time. Ideally, the *web 2.0 mashups* course is done in combination with the second year CS **data-driven applications**, that treats data management issues and programming aspects.

Nevertheless, to present the information about mashups, the instructor(s) must take an **example-based approach**, such that the students can base their work on extending one of a well-selected **set of examples**.

afterthought(s)