Leveraging XML Technology for Web Applications

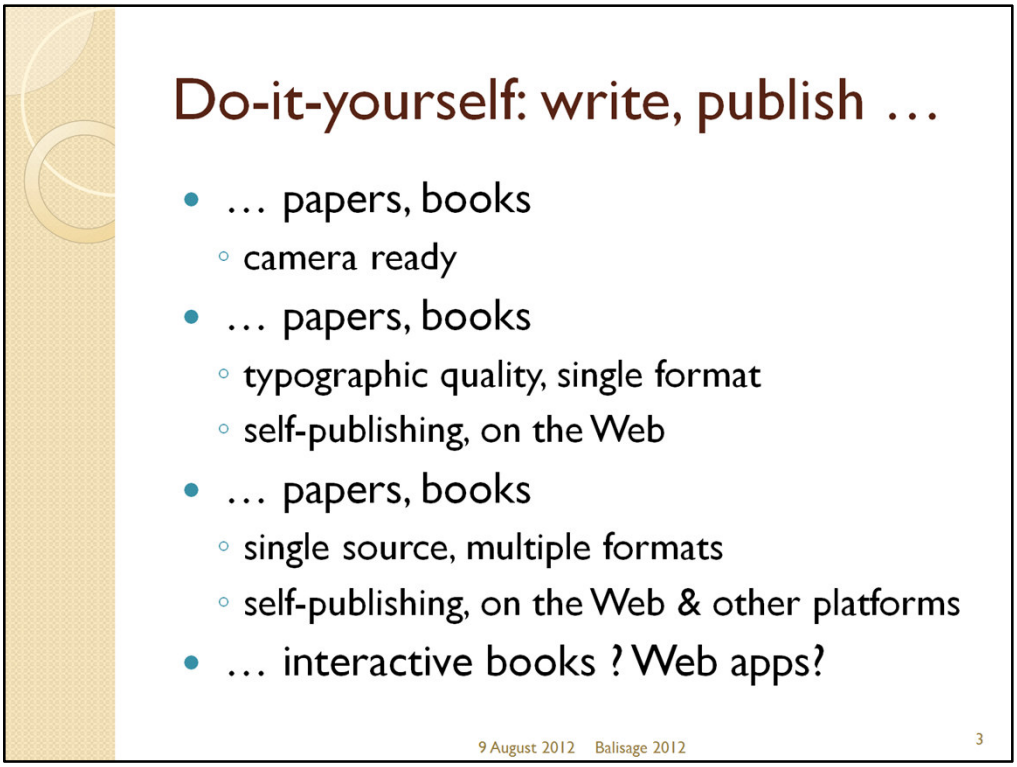
Anne Brüggemann-Klein

Joint work with:

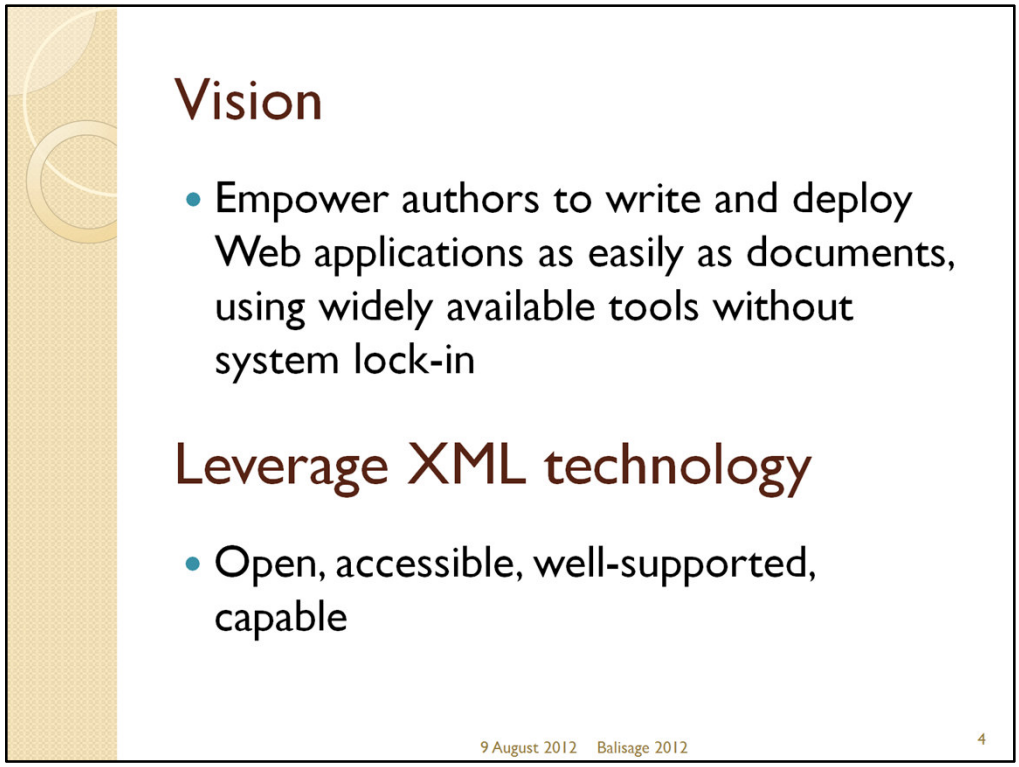
- Jose Tomas Robles Hahn
- Marouane Sayih

Technische Universität München

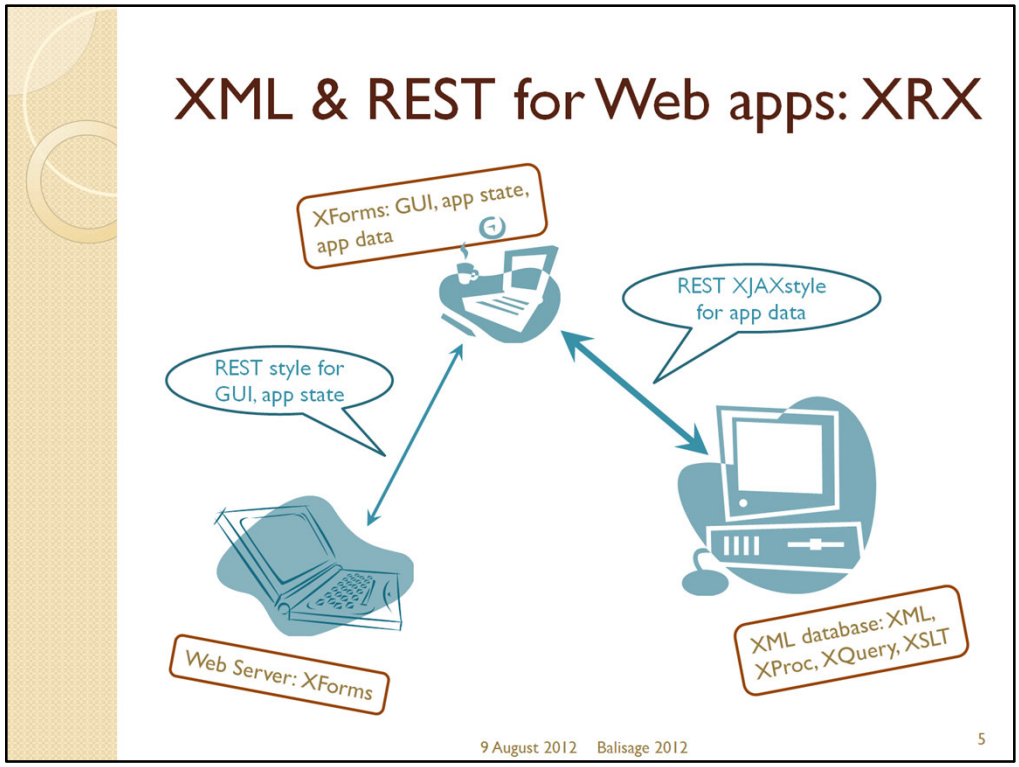
My personal document saga. Pushing boundaries, evolving technology. Now boundaries between documents and software are blurring.



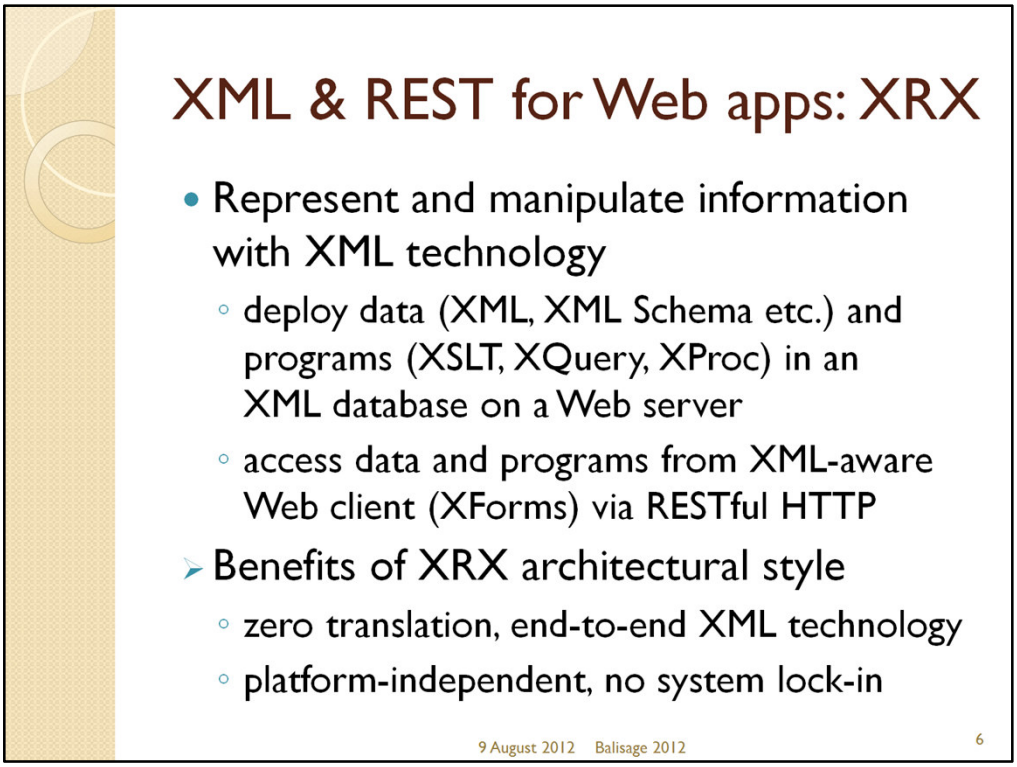
My personal document saga. Pushing boundaries, evolving technology. Now boundaries between documents and software are blurring.



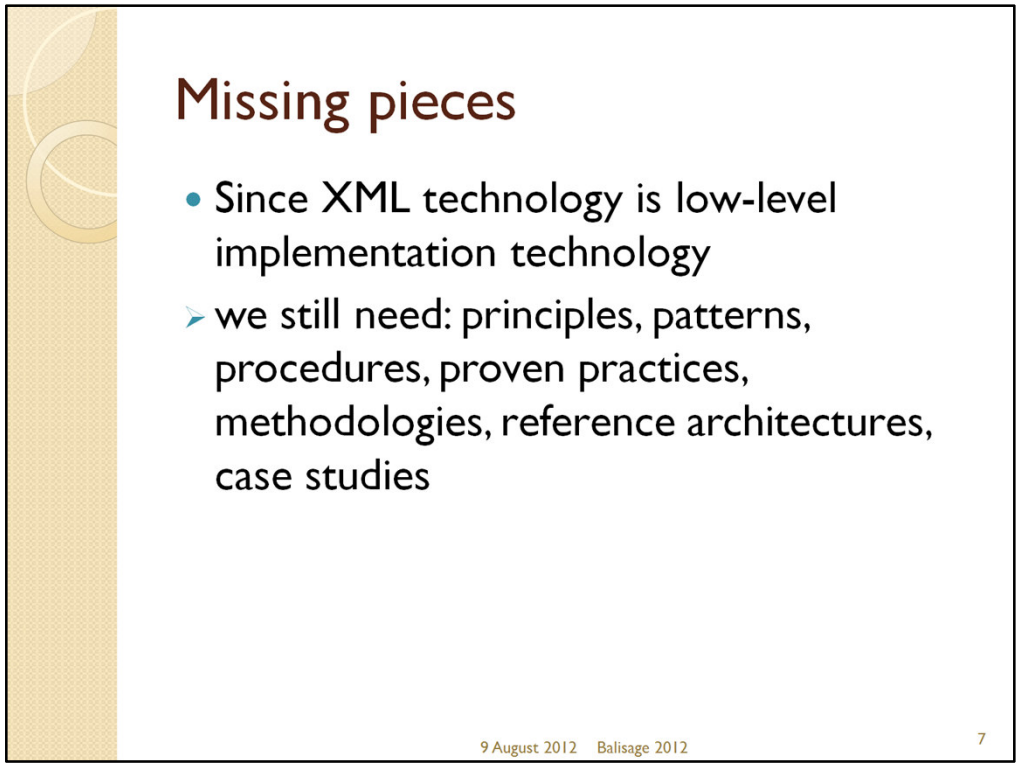
We as authors have experience as programmers: Makros, transformations, stylesheets, queries. Can we leverage our technology to target Web apps?



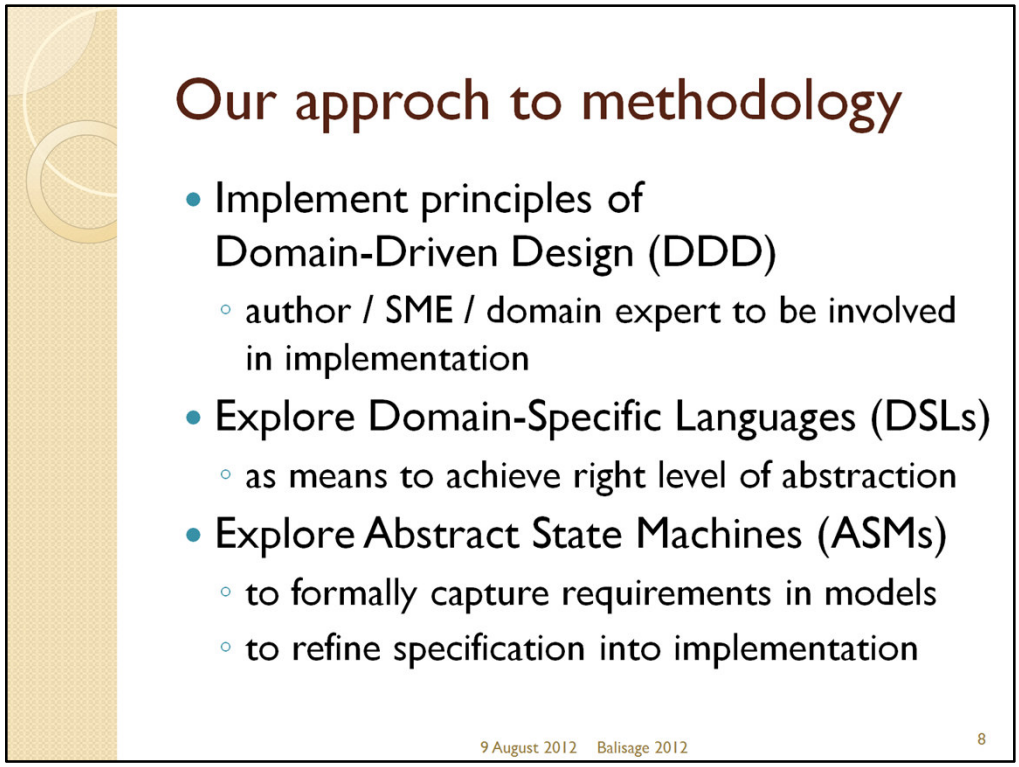
XForms as an application language, is that the right level of abstraction? As someone who has experience with writing layout algorithms in TeX‘s makro language I would not trust in Turing completeness alone. But if we can delegate the core programming to XQuery, XSLT oder XProc, we are in business. This is where REST comes in.



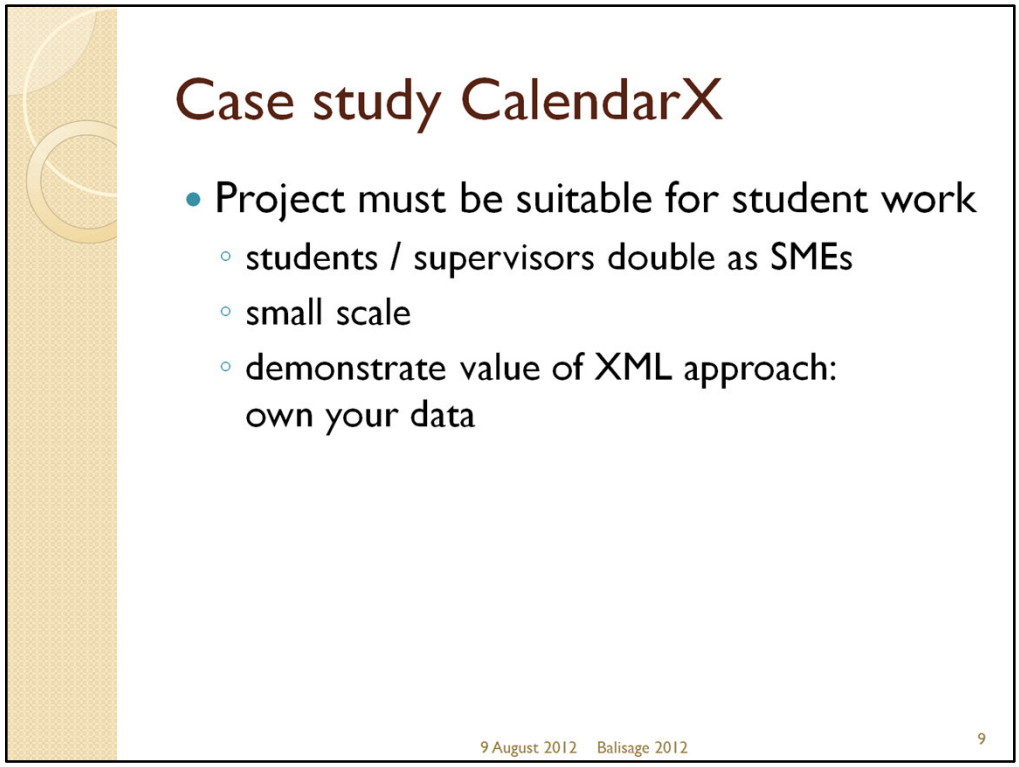
XForms as an application language, is that the right level of abstraction? … Zero translation, end-to-end XML: Nice for your data if they can stay in the model that fits them best, but it is also nice for you all who know how to program with XML languages, because you can use your expertise. … Platform-independence: Take claims with a grain of salt, for example some protocol for XForms processor and possibly other tasks such as URI mapping.



I know I am going overboard with the „P“s here, even beyond the „Triple-P“ paper we had in EML 2007, whose title was inspired by the „Triple-P“ parenting method. As to stealing, or letting oneself be inspired, or relating different areas: Of course, none of the ideas presented here are completely new. Software Engineering offers a rich source of knowledge that can be adapted to this domain of Web applications with XML technology. And these keywords have guided software development from the level of programming skills to an engineering discipline.



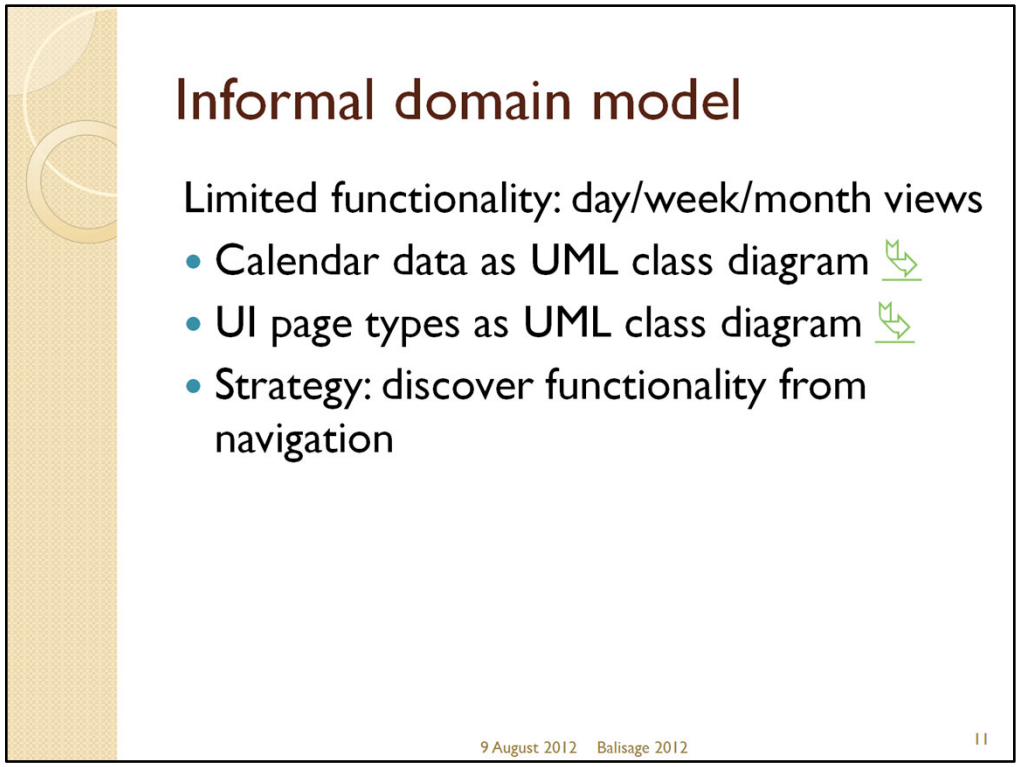
DDD originally to handle large and complex software projects. Here to empower domain experts to write their own software. DSLs can be specified and even be implemented by SMEs, e.g. schemas; or they serve as interface, a level of abstraction that encapsulates technical aspects that require programming expertise. ASM for formally specify and step-wise refine into implemenation, in contrast with Model-Driven Architecture (MDA).



Problems with calendar system since Palm organizer gave up on me. It annoyed me no end to have to rely on some kind of software magic to transfer data to a cheap mobile (LG) and to deal with severe restrictions as to data space and functionality, for example not being able to delete appointments.

Case study CalendarX

- … with a twist: rich data model
- model these as a single, recurring event that can be handled as a single unit:
  - class EP meets Tuesdays from 10 to 12 am and Fridays from 9 to 11 am during summer term 2012, but not on public holidays and not during the week I am at a conference
  - committee X has six meetings that are irregularly scheduled at different times of day and in different rooms



I am taking you step-by-step through analysis, design and implementation of CalendarX, discussion points of methodology on the way. We are only doing this for a simple subset of CalendarX functionality, namely to provide views of calendar data by day, week and month. Since we have a rich model of events, this is not as trivial as it may seem. … The domain model is key in our approach. We start with an informal description.

Informal domain model

- Cycle of navigation, under user control
  - each page when visited builds itself from
    - nextCalendar: Calendar
    - nextDateInfo: PartialDate

    computing type-specific date info and events
  - it then allows the user to specify
    - nextCalendar: Calendar
    - nextDateInfo: PartialDate
    - nextPage: {dayView, weekView, monthView, quit}

    and signal completion

Informal domain model: functions

DayView, WeekView, MonthView

- events(): Event*

Calendar, SuperEvent, EventRule

- getEventsForDay (d:Date): Events*
- getEventsForWeek (firstD: Date): Events*
- getEventsForMonth (m:Month, y:Year): Events*

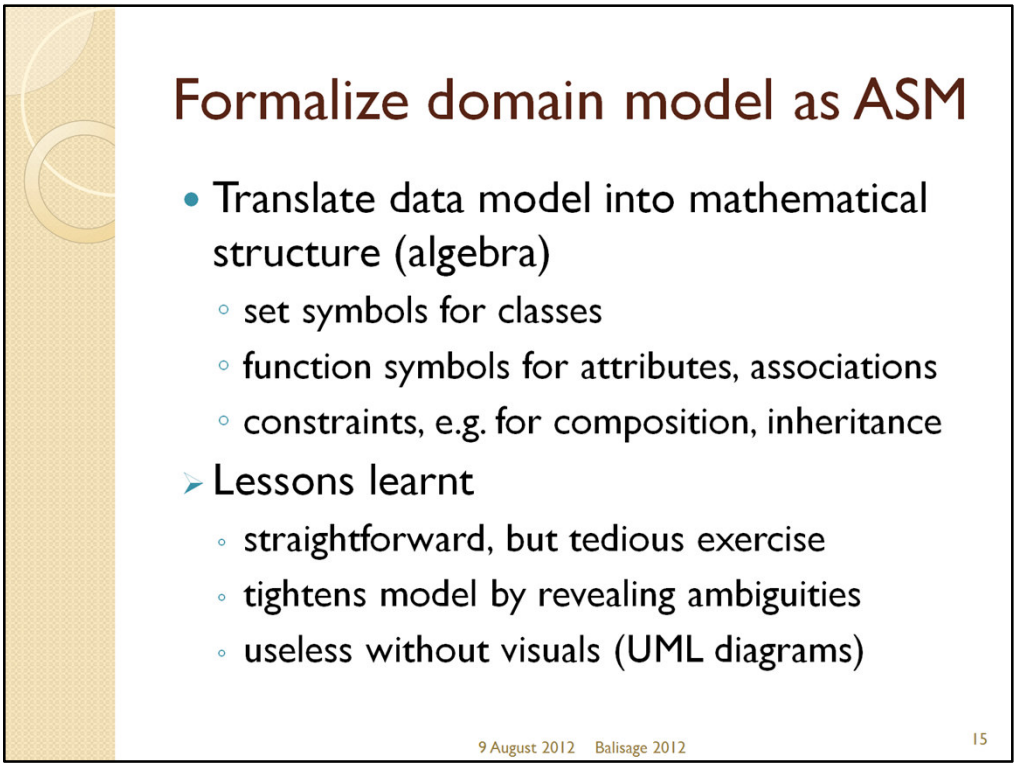
Pattern

- matches (d:Date): Boolean

Informal domain model: functions

Global

- datesForWeek (fD:Date): Date*
- datesForMonth (m:Month, y:Year): Date*



Useless without visuals: You cannot start with ASMs as a method of requirement elicitation. Domain experts won´t buy it.

Formalize domain model as ASM

- Translate cycle of navigations into ASM computation
- Lessons learnt
  - for this case study simple, straightforward
  - very small ASM program that delegates complex functionality to static functions that have to be specified separately
  - good basis for architecture

  probably typical for Web applications

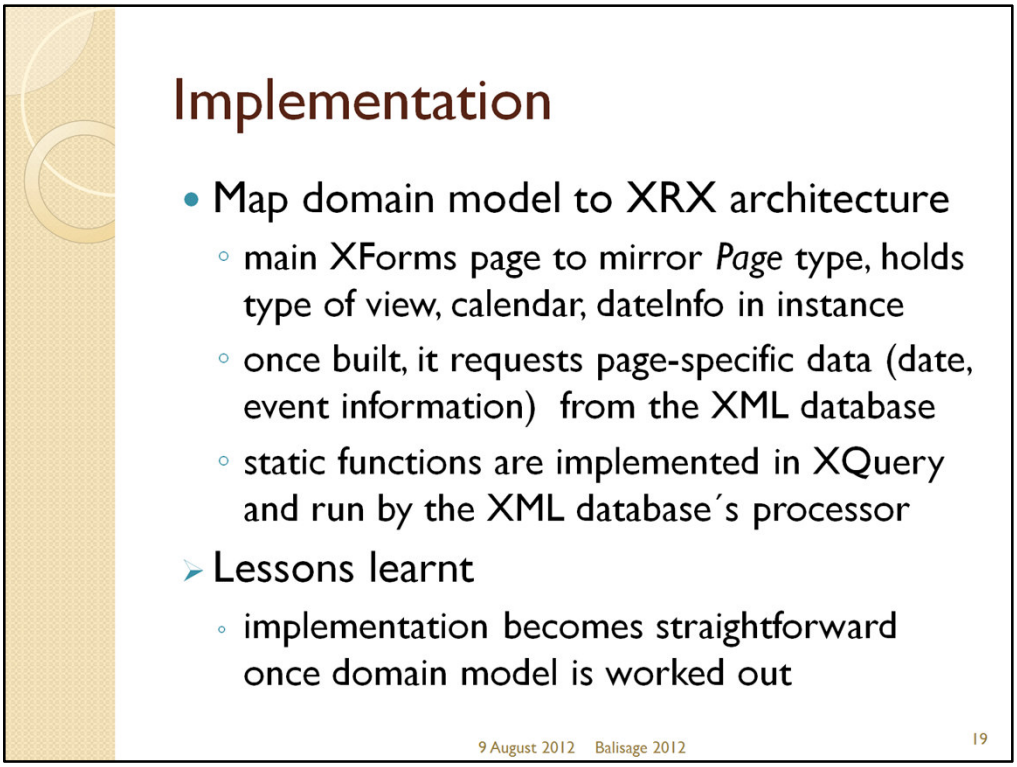
Formalize domain model as ASM

- Specify static functions as pseudocode programs or mathematical expressions (out of scope for ASMs)
- Lessons learnt
  - straightforward, low complexity
  - valuable basis for implementation in XQuery, which is a straightforward translation

  presumably the most useful part of this exercise, yet outside the core of ASMs

Implementation

- Translate conceptual model of calendar to XML Schema, building on previous work (EML 2007, Balisage 2009)
- Lessons learnt
  - systematic transformation prefered over automation
  - the schema is a useful DSL but could and should be streamlined via a metamodel (Balisage 2010) for domain experts that are no XML Schema specialists



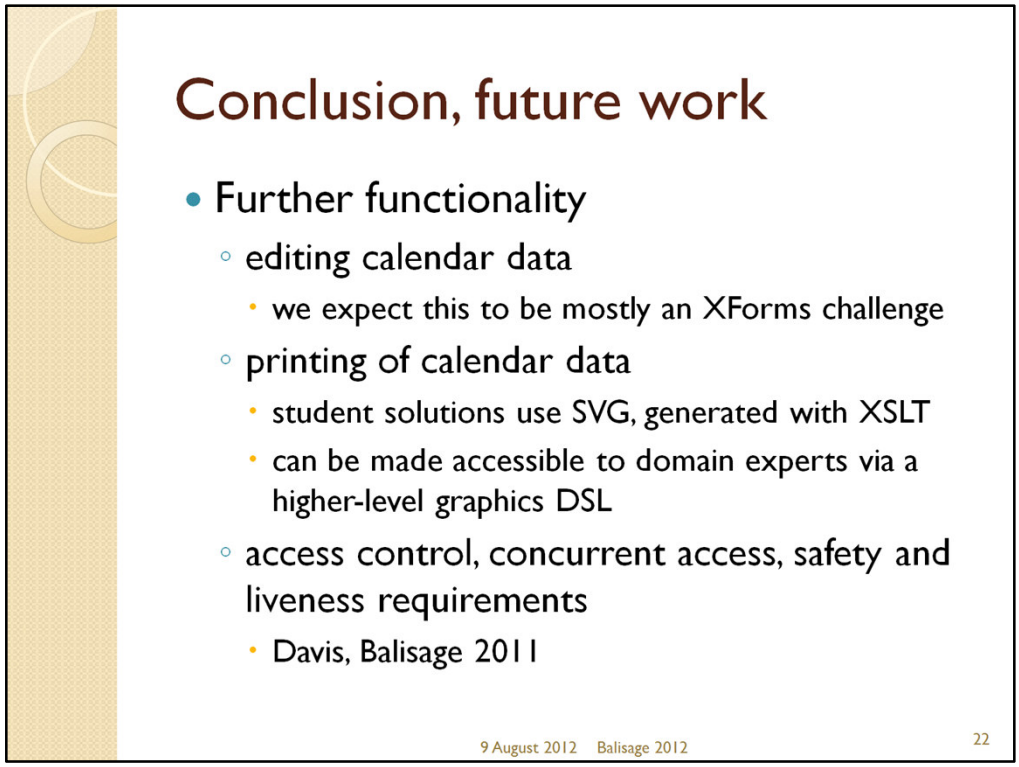
Also beneficial for student instructions. Takes the mystery out of student solutions. Am even doing an implementationi myself.

Status

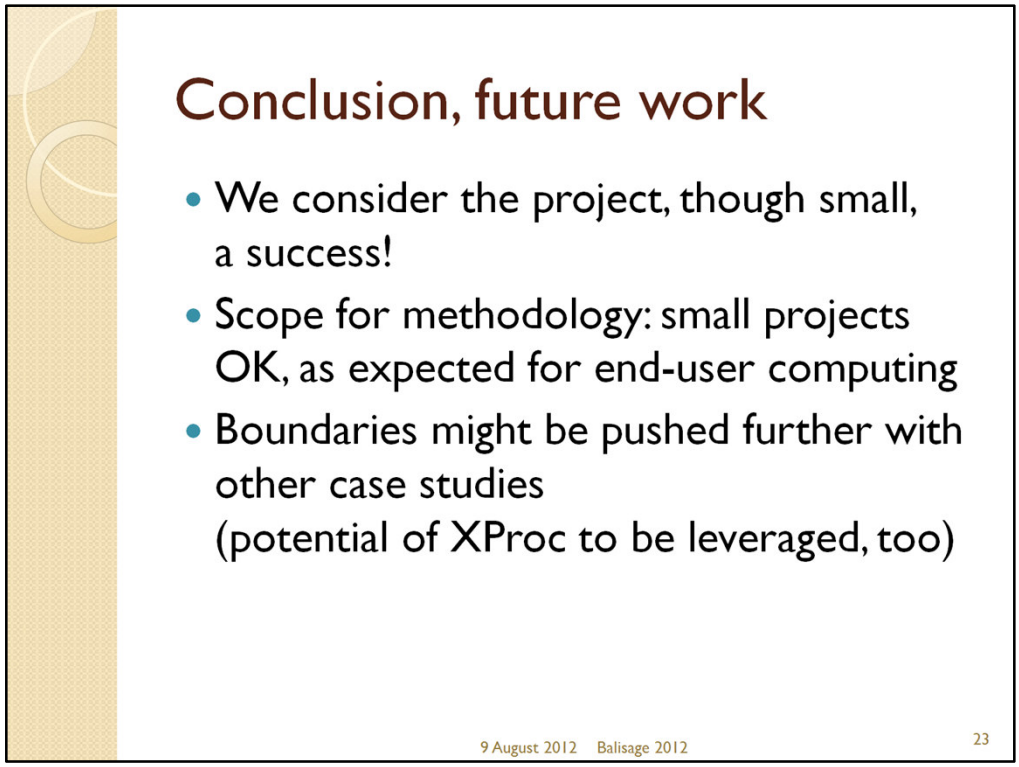
- Several implementations of CalendarX on the basis of the informal Domain Model exist (MTh Robles Hahn, several groups in lab course XML Technology)
- Platform used: Firefox browser, Orbeon Forms XForms processsor and eXist XML database in Tomcat container

Conclusion, future work

- The implementations are largely platform independent (Orbeon Forms, XSLTForms), due to exclusive use of XML technology
- Formal specification and systematic derivation of implementation make building CalendarX straightforward
- DDD principle has been fully validated
- ASMs have been useful but not mission-critical: we will explore them further



Printing calendar data, such as monthly overviews: nice to have / carry around. And there is the DSL challenge.



Scale: digital edition of the Oxford English Dictionary OED has not been an exercise in personal publishing either.