State Chart XML as a Modeling Technique in Web Engineering

Anne Brüggemann-Klein
Marouane Sayih
Zlatina Keskivov

TU München

Balisage 2014

Agenda

- XML technology for Web applications: motivation and background
- The simple case: event – activity paradigm
- Requirement to support event – state – activity paradigm
  - statecharts and their encoding State Chart XML (SCXML)
  - their role in the model-view-controler architecture
- Conclusion / further work

Why you might be interested

- You use XML technologies as a domain expert doing "end-user programming"
  - ☑ you code information in XML (or deal with others' encodings)
  - ☑ you model data with XML schema languages (or …)
  - ☑ you query and transform data with XSLT / XQuery
  - ☑ you configure XML editors with XForms
  - ☑ you configure XML processes with XProc
- You are ready to explore the "adjacent possible"
  - ☑ your code may have an "improm pu" character, but is worth keeping
  - ☑ you wish to bundle XML stuff into an application while staying within a "lightweight" XML technologies framework
  - ☑ you may wish to make it ready for the Web (while …)
  - ☑ you wish to maintain an adequate level of software quality: sustainable (conceptually sound, useful, comprehensible, extensible)

XML technology is ready for just that !

- History of exploring and teaching XML technology
  - student lab courses and student projects with XML technology: XBlog, CalendarX, GameX
- Two premises for accessibility and software quality within the realm of XML technologies
  - methodology of domain-driven design
    - uses explicit, technology-independent domain model throughout the development process including implementation
    - encourages collaboration between technical and domain experts (or lets domain experts go it alone ☺)
  - use XML technology not only for implementation but modeling, too
    - new for ➢ modeling "behaviour" of applications: !! statecharts !! for visualization and !! SCXML !! for encoding

Scenario and first example: → GuessTheNumber

- A bunch of queries, transformations, visualizations, forms for some related tasks
  - impromptu character, but worth keeping (und thus worth doing well, within limits)
  - bundle into an application, so you make explicit (and don't need to remember) the parameterizations, the constraints, the workflow …
- Might be easiest to leverage Web technology and request-response cycle
  - run your functions on a Web server, for example on a (local) XML database
  - access the functions by URL from a Web browser in a coordinated manner, providing parameters through a form (HTML forms, XForms when transmitting XML data)
  - return a form to the browser

How is GuessTheNumber organized ?

- A central query play.xq
  - running on eXist in a Web server
  - called from a browser with parameters cmd (hidden value) and arg (explicit value set by user) provided by a form
- The query play.xq
  - is triggered by browser requests and handles them
  - decides what game function it needs to call
    - compute secret number within range
    - evaluate guess (high / low / correct)
  - computes the next screen for the user
- The query play.xq controls game functionality (model) and user screens (view) in a request-response cycle → Web-adaptation of architecture Model-View-Controller (MVC)

What skills are required ?

- Take your "model" queries and copy them into database
  - figure out how to compute the logarithm of the range value in XQuery for a better estimate
    for number of tries … that's the most challenging here …
- Set access rights
- Figure out how to store XML data for game data between calls to server and how to load them (XQuery module)
- Figure out how to access query parameters (XQuery module)
- Figure out how to send HTML response (just return it)

➢ all within your domain of expertise !

A more general case

- Why is GuessTheNumber so easy to implement ?
  - controller just needs a simple map from user events to activities (no history information, no context information)
  - ➢ "event – activity" paradigm
- In the more general case
  - controller needs to maintain state information: what sequences of events are mapped to the same / different activities ?
  - ➢ "event – state – activity" paradigm
- Now we are in the danger zone
  - dealing with state in applications in an ad-hoc manner is a receipe for desaster (beware of GUI editors !)
- Are there modeling techniques to the rescue ?
  - Yes: statecharts (encoded by SCXML)

Model-View-Controller revisited

Division of labor between MVC components

☑ Tasks of view (HTML forms, XForms)
- present information to the user
- offer means of interaction to the user
- pass on user events to controller via HTTP requests

☑ Tasks of (passive) model ("small" queries)
- manage persistent data of application
- provide domain functionality to potential clients

⁉ Tasks of controller (request-response cycle)
- ☑ be triggered by user events from view
- ⁉ interact with model, based on history of past user events (state)
- ☑ tell view which information to present / which interactions to offer to the user via HTTP response

Statecharts …

- Modeling technique proposed by Harel to model the "behavior" of event-driven systems (that react to signals)
  - behavior: control the activities of a system in response to outside events → "C" in Web-adapted MVC
  - a diagrammatic language that has a formally defined semantics and that can be executed (providing a simulation of the system)
  - considerable enhancement of basic state diagrams
- Basic state diagrams: notion of states and transitions between states, triggered by events

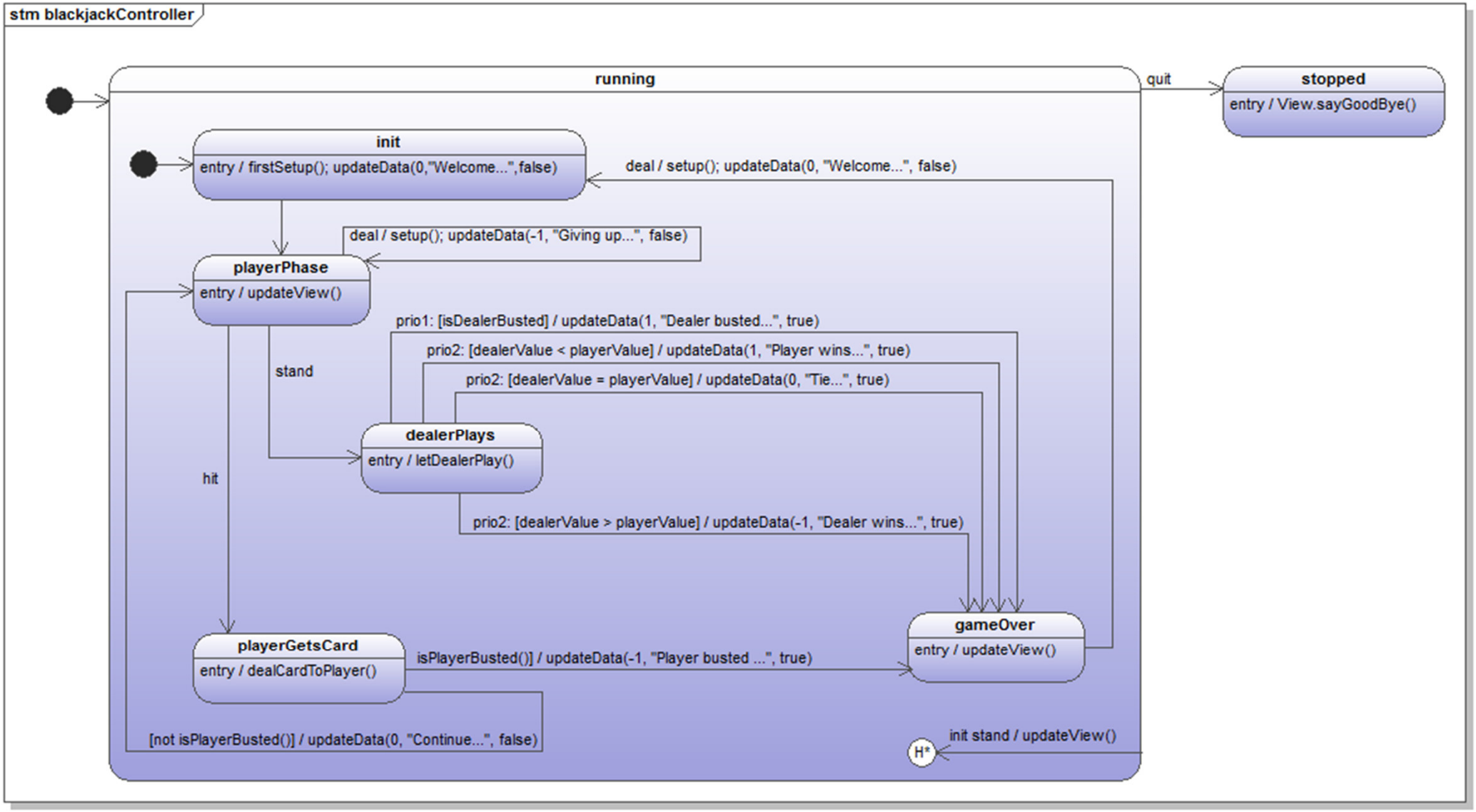
+ Activities to be executed when entering or leaving a state and when taking a transition: manipulate data, raise internal events, call external services, log information, …
+ Guard on transitions: events AND / OR conditions

… Statecharts

- Hierarchy of states
  - used for decomposition (top down)
  - used for abstraction, default and overriding behaviour (bottom up)
- Parallel states
  - used for concurrent (but synchronizable) activities
  - used for orthogonal decomposition of state
- History states
  - used for (single-level) backtracking
- ➢ no state explosion !

Statecharts by example

- Another case study: → Blackjack
  - student project in Coursera MOOC on interactive Python
    - in "Fundamentals of Computing" series from RICE University
    - speaking of end-user computing, as an aside:
      course comes highly recommended, if someone wants to upgrade their computing skills, in spite of viability of end-user computing with XML technology ☺
  - requires "event – state – activity" paradigm in controller



Generated by UModel www.altova.com

What is that statechart good for ?

Basis for reflection on / reasoning about the design

- Quality of structure (clear, natural, level of abstraction) ?
- Verification
  - Have we represented the rules of blackjack correctly ?
  - Have we covered all combinations of potential user interactions ?
- Guide for implementation of controller (in XQuery)
  - list of activities in controller statechart
    → requirement analysis for view and model capabilities
  - systematic derivation of implementation: table of activities, indexed by state / event and conditional to guards
- Enables code generation (with XSLT)
  - when encoded in SCXML (which is straightforward)
- Point of reference when changing / extending functionality

Further case study: GameX

- GameX (presented at Balisage 2014)
  - a serious browser game to further systemic thinking in players
  - implemented with XML technology
- Statechart (→ diagram and SCXML encoding) for gameplay rules of GameX as part of MTh Keskivov
- Work in progress: modeling the gameplay of GameX, with the following goals
  - modular design that lends itself to decomposition for joint project in lab course
  - explicit strategy for data management
  - explicit consideration of concurrency
  - verifiable consistency of data

Conclusion

- Our "hobbyhorse" at EPT:
  Web applications with XML technology
  - XML languages and tools for implementation
  - XML languages and tools for modeling and design
    - previously: modeling data; now: modeling behaviour
- XML technologies are viable for end user programming
  - current practice: domain experts …
    - code information in XML
    - model data with XML schema languages
    - query and transform data for viewing etc. with XSLT/XQuery
    - configure XML processes with XProc
  - explore the "adjacent possible": domain experts …
    - bundle their code into applications and put them on the Web

Relevance

- End-user programming and Big Data
  - getting XML data into analysis tools !
  - promoting XML as "lingua" franca for data to be analyed?
  - implementing applications for XML data ?

Current / further work

- A conceptual model for SCXML statecharts
  - NB: SCXML is based on Harel statecharts, but there are a number of differences (cf MTh Keskivov)
  - further variants of statecharts: Harel OO, UML 2.0: comparison in progress with Sayih / Alawadai
  - diagram of concepts and mapping to SCXML encodings
  - definition of semantics in terms of the conceptual model
    - tricky in the presence of nested parallel states
  - do we need / want / have a concept of sub-statechart
- Discover / develop good practices for statecharts
- Execution of statecharts (possibly with UML tools ?)

Resources

- Literature
  - Harrocks: *Constructing the User Interface with Statecharts*. Addison-Wesley 1999.
- Tools
  - SCXMLGUI
  - Altova UModel