# Chapter 8

# Conclusion

## 8.1 Related Work

### 8.1.1 Server Pages Technologies

Server side scripting technologies, like Active Server Pages (ASP) [110] [147], Python Server Pages (PSP) [4], and Java Server Pages (JSP) [143] allow to embed script code into HTML, or more generally XML. The Java Server Pages Standard Tag Library (JSTL) [51] improves JSP technology by providing tag support for typical actions. None of these technologies offer NSP's static type checking.

### 8.1.2 The "System Calls User" Approach

The language Mawl (Mother of All Web Languages) [7][8][106] is a server script approach with a sophisticated concept for form presentation. The definition language for forms is an HTML dialect. The scripting language is oriented towards the programming language C. Form presentation to the user is done in a "system calls user" way, i.e. a procedure is called in the server script. The actual parameters that are passed to the procedure are the data presented to the user, the return value of the procedure is the data entered by the user. This way Mawl allows for seamless integration of a script organizing session concept. The input type and output type of a form must be declared. Mawl supports static type checking of these types. Iterator tags, indexing for list data, and a dot notation for record data is available in the form description language for data presentation. Mawl separates between service logic code and markup language from the outset. Based on this device independency [23] is introduced to a certain extent: forms cannot be described in the aforementioned HTML dialect only [6], but in the Phone Markup Language[1] (PML) [151], too. In the "system calls user" approach the overall control flow is prescribed by the server

[1] PML allows for describing documents that are served over a telephone by special telecommunication portal middleware. PML is one of the precursors of Voice XML [118].

script code, because the return point of a form presentation is fixed. Form presentations are treated like procedure calls: after form processing the control flow returns to the call point. This abandons the core paradigm of hypertext, where a page may encompass several links and forms each targeting different locations. In order to add flexibility, Mawl supports multiple submit buttons. In the case that a form should be able to target different server actions, the receiving script can analyze, which submit button has been pressed, and can branch the control flow appropriately [44]. That is the purpose of usual HTML multiple submit buttons, too. However the resulting designs relying on such control flow branching must be considered flawed. These systems seriously suffer an "ask what kind" design antipattern resulting in high coupling and low cohesion. The situation is even worse, if multiple forms should target different server actions. These multiple forms must be emulated by one single superform and several submit capabilities. The output type of the superform must encompass the output types of all emulated forms, which has to be considered a further design flaw. We use the term "ask what kind" as a antipattern name in a generalized sense. Assume that a form containing several submit capabilities should be reused. Then every scripting code that reuses the form is responsible to branch the control flow correctly - high coupling. This is an error-prone pattern. Furthermore now consider a change to the submit buttons in the forms. Now it is necessary to be aware of all the distributed hard-wired case structures and to fix them accordingly - low cohesion. The term "Don't ask what kind" [36][37] has be coined in the object orientation community for usage of polymorphism as a basic pattern of assigning object responsibilities [107] for similar reasons.

The project Bigwig [18] declares itself as an intellectual descendant [15] of the Mawl project. Bigwig provides higher order templates [17][157]. However the motivation for higher order templates in Bigwig is different from the motivation of higher order server pages in NSP. In [18] gaining the flexibility of print-like statements in script-centered languages is given as a reason for the concept of higher order templates - in Bigwig no iterator tags are used in form descriptions as in Mawl. In contrast, in NSP higher order server pages are motivated very targeted as an enabling technology for maintainable solutions for typical design problems. Bigwig uses the non-standard notions of gaps and plugs for formal and actual parameters of form templates. A formal definition based on data flow analysis [131] is given for the type system of the language DynDoc, which is a sublanguage of Bigwig [17][157]. The defined static semantics guarantees user interface description safety. The Powerforms technology [16] is part of the Bigwig project. Powerforms defines a declarative language for expressing form input field formats and interdependencies between form input fields. Powerforms allows for the generation of client side code for ensuring the declared constraints. There is also a Java-based successor to Bigwig, called JWIG [31][32].

The rule-based CGI programming language Guide [109] supports a template mechanism that is similar to the one of Bigwig. However Guide does not support static type checking.

### 8.1.3 Web Application Frameworks

The NSP approach must not be mistaken to be a variant of architectural approaches. These web application frameworks target separation of presentation, content and logic to different extent, whereas they follow an approach to web application architecture that has become known as Model View Controller (MVC) architecture or Model 2 architecture. Further objectives of these web application frameworks can be integration with enterprise application frameworks, rapid development by providing an integrated development environment, support for internationalization, support for security, or support for dynamic form field validation. A prominent commercial MVC web application framework is the Sun ONE Application Framework (JATO) [167]. The most widespread application servers according to [149], namely Oracle9iAs, IBM WebSphere, and BEA WebLogic, come along with MVC web application frameworks, too. WebMacro [84] is an early open source project that allows the separation between Java and HTML. Other prominent open source web application frameworks are Struts [50] and Cocoon [39] - both hosted by the Jakarta project. Wafer (Web Application Research Project) [175] is a research project that collects, investigates and compares existing open source web application frameworks.

### 8.1.4 Web Services

Web Services [178] are widely regarded as a major technological shift in the usage of the web. Web services are primarily discussed in the B2B domain. Web services possess a type system, the Web service definition language [33]. At first glance, web services may seem inspired from and similar to user interfaces, using the same protocol, namely HTTP [14]. With respect to the NSP static type checking we can identify a clear difference between web services and HTTP/HTML user interfaces: in web services there is no necessary type relation between different messages, although there may be a type system. The types of different messages can be chosen freely according to the needs of the business case. In HTML dialogues however, if the user is supposed to send a form with data to the server, then a page containing this form must have been previously sent to the user. More generally speaking, we have a necessary relation between a typed user request and a systems response before that request. This relation is based in the mechanism itself.

In other words, in HTML dialogues, the type information is not transmitted once and used for all subsequent interactions, but it is transmitted before every typed request, and it is directly transformed into the displayed form, which the user actually fills out. This is not an accidental design mismatch between web services and HTML/HTTP dialogues, but a fundamental difference caused by the fact that web services are accessed automatically, and HTML dialogues are designed for end users.

### 8.1.5 XML Technologies

The XForms standard [69][68] defines the successor to HTML forms. XForms offers tag support for gathering structured data as XML documents. Data are transported as XML. In XForms data is separated from presentation from the outset, furthermore the XForms controls has to be considered platform independent, i.e. device independency is targeted. It is possible to specify constraints on data gathered by a form. Thereby the XForms approach does not only allow constraints on type correctness and required data entry, but allows to specify arbitrary validations and calculations with respect to data entered by the user. An event model defines hooks for handling violations of the specified constraints. In order to address parts of XML data in expressions XForms relies on XPath [35]. In HTML/XHMTL only plain unidirectional links are possible. Though a limited number of different link behaviors is available, important simple notions like e.g. a desired page decomposition are not defined for HTML/XHTML. XLink [52] overcomes these limitations in a general XML setting, whereas it has been influenced by [99]. XInclude [114] improves XLink with respect to document decomposition by defining a processing model for merging documents.

### 8.1.6 Functional Programming Language Approaches

The comprehensive WASH project offers the two technologies WASH/CGI [170] and WASH/HTML [168][169] for web authoring based on the functional programming language Haskell. WASH/CGI offers static checks for user interface type safety, WASH/HTML offers static checks for a weak user interface description safety. WASH/CGI is a domain specific language for server-side Web scripting. Dynamic generation of HTML output is monad-based. The form presentation concept encompasses a callback mechanism that allows for full design flexibility. WASH/CGI programs are dynamically type-safe: untrapped errors cannot occur, because necessary server-side reaction to user entered data that cannot be parsed is enforced. A compositional approach to specifying form elements is offered. WASH/HTML defines a Haskell combinator library for dynamic XML coding. For WASH/HTML systems full XML validity but only a limited form of HTML/XHTML validity can be guaranteed statically. In [120] a Haskell library for CGI scripting is proposed. There is also a server pages technology available for Haskell [122]. Haskell Server Pages guarantee well-formedness of XML documents. The small functional programming language XM$\lambda$ [162] is designed to ensure full XML validity [121]. XM$\lambda$ is based on XML documents as basic datatypes. The approach given by [120][122][162][121] comes along with a client side scripting technology for Haskell [119]. Yet another project in the context of Haskell is [180], which investigates two different approaches. In the first approach a library for XML processing arbitrary documents is provided. Thereby well-formedness of XML documents is ensured. The second is a type-based translation framework for XML documents with respect to a given DTD, which guarantees full XML validity.

The LAML (Lisp Abstract Markup Language) project [133][134][135] proposes web programming based on Scheme. Two other Scheme based web publishing systems are proposed in [148] and [81]. Beyond this it is shown in [81] how the features of a Sheme extension can be exploited for an efficient web server implementation. A conceptual basis for the continuation-style of functional web publishing technologies is given by [94] as a generalized notion of monads.

### 8.1.7 Logic Programming Language Approaches

In [86][87] a mature CGI programming library for the functional-logic multi-paradigm language Curry is proposed. A notion of submit button event handler is introduced and enables in effect a callback-style programming model. A specific abstract data type is used for references to input fields, which enables compile-time checks with resepct to input field naming. Pillow [25][24] is a CGI library for the CIAO Prolog system. LogicWeb [161] is a toolkit for even improved amalgamation of logic programming and web programming. Neither the Pillow approach nor the LogicWeb approach offer user interface type safety.

### 8.1.8 Web Application Reverse Engineering

The tool presented in [90][89][88] analyses source code and pages of a web application and generates an architecture diagram that visualizes the interactions between static pages, active ASP or JSP pages and other software components by arrows. The same support is offered by the tool WARE (Web Application Reverse Engineering) [112] for ASP and PHP based systems. A technique for recovering navigational structure and a conceptual model from a web application without tool support is described in [5]. An example for a tool that can track the change history of a web site is given in [22].

## 8.2 Further Work

#### Formal Parameter Requirement Specification

In the NSP approach it is possible to place requirements on the single formal parameters of web signatures. It is possible to require data entry by the user, a widget or a special kind of widget. The fulfillment of the requirements is statically ensured, thereby the NSP active controls are enabling technology. The possibility to place requirements on formal parameters can be generalized to arbitrary constraints. A suitable declarative constraint language can be defined, from which client-side checks are generated. The approach can be elaborated for array parameters.

#### Improved Design Recovery

The information extracted by JSPick from a Java Server Pages based system can be analyzed with respect to several classes of potential sources of error as

already explained in 6.1. The amount of extracted information can be improved, i.e. lower and upper bounds for generated controls can be inferred.

#### Integration with Generator Type Safety

The notion of generator type safety is defined for a generic programming [47] mechanism that integrates parametric polymorphism and static introspection. The new generic programming mechanism has been proposed in [64]. The mechanism is powerful enough to provide simple and intuitive solutions to typical crosscutting problems, like e.g. transparent data access layers [9], as reusable components. Thereby the proposed balanced mechanism allows for static checks of generator type safety. The mechanism outlined above can be integrated in the NSP server pages concept. As an example, with the resulting technology it will be possible to implement a generic data form: given a class description a generic include server page generates a form with input capabilities for all attributes of the class.

## 8.3 Summary

The NSP project investigates client page description safety and client page type safety with respect to server pages technology. The contributions are

- parameterized server pages,
- higher order server pages,
- server-side calls to server pages,
- direct tag support for user defined programming language types in writing forms,
- virtual exchange of programmed objects across the web user agent,
- dynamically type safe direct input controls, statically ensured automatic constraint checks,
- introduction and formal definition of a tool for code-structure sensitive recovery of web signatures and form types from server pages based presentation layers,
- formal definition of static client page type and description correctness as a Per Marin-Löf style type system,
- and the integration of a scripting technology with a modeling technique.

Thereby the results are programming language independent.