compte-rendu intermédiare

Projet ANR-09-EMER-009-01

PWD/Programmation du Web Diffus

Programme Emergents 2009

A Identification

Acronyme du project	PWD
Titre du project	Programmation du Web Diffus
Coordinateur du project (société/organisme)	INRIA Méditerranée
Date de début du projet Date de fin du projet	Novembre 2009 Octobre 2013
Labels et correspondants des pôles de compétitivité (pôle, nom et courriel du corresp.)	Systématique
Site web du projet, le cas échéant	`http://www-sop.inria.fr/indes/pwd`

Rédacteur de ce rapport	
Civilité, prénom, nom	Manuel Serrano
Téléphone	`04 92 38 76 39`
Courriel	`Manuel.Serrano@inria.fr`
Date de rédaction	Juin 2012
Période faisant l'objet du rapport d'activité	Août 2011 - Mai 2012

B Livrables et Jalons

No	Intitulé	Nature*	Date de fourniture			Partenaires
			Initiale	Replanifiée	Librée	

La période T0+30 ne correspond à aucun délivrable concret. Néanmoins nous progressons régulièrement sur nos divers chantiers, ainsi nous n'anticipons pas de retard sur les tâches suivantes :

- 3.2: *Multimemdia and Home automations APIs* (hormis le fait que comme annoncé lors de l'évaluation à mi-parcours, nous avons jugé pertinent d'opérer un glissement de la domotique vers la robotique).
- 3.3: *Service Discovery*

C Rapport d'Avancement

C.1 OBJECTIFS INITIAUX DU PROJET

Le Web est la nouvelle plate-forme où seront déployées les applications de demain. Bien qu'aujourd'hui déjà large, le Web ubiquitaire va encore s'agrandir pour connecter tous les nombreux équipements électroniques qui nous environnent. Mais bien qu'ayant permis l'émergence d'applications novatrices telle que Google Map, le Web requiert de penser autrement la programmation.

Notre réponse à ce défi repose sur plusieurs principes directeurs. Une application pour le Web n'est plus une constellation de pages dynamiques mais un unique programme cohérent, projeté sur les serveurs et les clients (éventuellement nomades). Une telle application agrège de nombreuses sources de données tout en étant dirigée par de multiples flux d'évènements.

Ces nouvelles applications pour le Web trouveront, par exemple, à s'exercer dans les domaines de la domotique et du multimédia. Elles requièrent d'étudier plus avant les langages de

programmation et leur sémantique, leur compilation sur de multiples cibles et leur écriture et mise au point à l'aide d'environnements de développement appropriés.

L'objectif principal du projet PWD, est de développer de nouveaux langages dédiés à la programmation de ce nouveau Web. Pour cela nos études porteront d'une part sur les fondements de ces nouveaux langages (sémantique formelle, typage, sécurité) ainsi que sur leur implantation (compilation efficace, système d'exécution, environnement d'exécution). Les langages ainsi développés seront validés par quelques applications Web innovantes complexes à écrire avec les langages de programmation traditionnels.

C.2 TRAVAUX EFFECTUES ET RESULTATS ATTEINTS SUR LA PERIODE CONCERNEE

Comme prévu lors de l'élaboration du projet PWD nous avons continué les développements des deux systèmes Hop et Ocsigen. Ces deux systèmes ont tous deux un rôle dual. D'une part ils constituent une finalité à part entière puisqu'une grande partie de recherches que nous menons sont motivées par des besoins ressentis lors de leur mise au point. D'autre part, ils constituent également une base concrète pour mener d'autres recherches plus prospectives. Par exemple les travaux menés autour de la sécurité des applications Web se fondent sur la théorie de la programmation multi-tiers qui est au coeur de nos deux systèmes. Dans la suite de cette section nous présentons d'abord les travaux que nous avons réalisés au cour de ces 6 derniers mois pour le système Hop, dans un second temps nous présentons ceux réalisés pour le système Ocsigen.

C.2.1 HOP

Comme spécifié dans la tâche (3.2) nous avons commencé la conception et l'implantation d'une API pour les applications multimédia. Cette API est disponible dans la version *2.3.0* de Hop. Nous l'avons principalement axée vers la manipulation des fichiers audio. Elle permet ainsi de décoder et encoder la majorité des formats courants (mp3, wav, flac, ...), d'extraire les méta données (tag id3), elle permet également de contrôler finement et efficacement les dispositifs matériels jouant de la musique. Ainsi, nous avons pu la déployer avec succès sur des équipements basse consommation qui peuvent être couplés avec du matériel HiFi afin d'obtenir de la sorte des chaînes HiFi distribuées. Nous étudions les suites à donner à ces réalisations. Nous envisageons soit une suite industrielle soit une diffusion sous forme de logiciel libre.

Comme nous l'avons annoncé lors de l'évaluation à mi-parcours, faute d'avoir trouvé des partenaires industriels, nous avons renoncé à poursuivre nos travaux dans l'axe domotique. Nous avons choisi de réorienter cette activité (tâche 4.2) vers la robotique. Nous pensons que ce domaine, qui est aussi très prometteur, sera, pour nous, plus accessible, en particulier parce qu'il ouvrira des opportunités de collaborations académiques. Ainsi, nous avons déjà créé une première API qui permet d'utiliser, depuis Hop, un certain nombre d'équipements robotiques normalisés. Cette API permet, par exemple, à une application Web d'avoir accès à toutes sortes de capteurs (gyroscope, accéléromètre, ...) et actionneurs (curseurs, interrupteurs, servomoteurs, ...). Nous avons pu réaliser quelques démonstrateurs et nous envisageons à l'avenir l'étude du contrôle de robots à câbles par des applications Hop. Ces applications consistent, pour une grande part, en une *orchestration* des divers équipements. Hop est actuellement mal adapté à ce type de programmation. C'est pourquoi nous nous sommes lancés dans l'ajout du couche réactive synchrone nommée HipHop qui sera décrite plus longuement dans les prochains rapports.

En marge de ces deux activités nous avons continué le développement de Hop proprement dit. Ainsi nous avons diffusé la version *2.3.0* qui embarque un nouvel interprète serveur, une nouvelle implantation du système de paquetage qui permet aux utilisateurs finaux d'installer des

applications Hop sans jamais quitter leur navigateur Web. Nous avons également commencé notre étude de la découverte de services et nous avons totalement finalisé l'implantation des événements serveurs (tâche 3.1).

C.2.2 OCSIGEN

Dans le cadre du projet Ocsigen, les 6 derniers mois ont été consacrés essentiellement au suivi de la sortie de la version 2 d'Eliom : amélioration des interfaces, de la documentation, réponse aux besoins des utilisateurs, amélioration de la compatibilité avec les différents navigateurs, test et amélioration des performances. Nous avons également poursuivi notre travail sur les server events (tâche 3.1) afin de les rendre plus fiables et plus souples d'utilisation. Le langage typé de description d'applications client/serveur (tâche 1.6) a vu quelques amélioration (notamment une plus grande souplesse dans l'utilisation côté client d'expressions définies dans du code serveur. Nous travaillons actuellement en collaboration avec la société BeSport (besport.com) pour étendre ce langage dans un cadre où un client souhaite communiquer avec plusieurs serveurs. Nous avons également repris le travail initié il y a quelques années sur la création d'un système de templates typé (tâche 1.5) : dans le cadre actuel où le Web devient plus une plateforme d'applications que de contenu, il nous semble de plus en plus clair que ce travail doit se rapprocher de la tâche 3.5 (user interfaces) sur laquelle nous continuons de progresser. Nous expérimentons actuellement la définition de widgets réactifs (natifs et en utilisant Google Closure ou JQuery).

C.3 DIFFICULTES RENCONTREES ET SOLUTIONS

C.4 FAITS ET RESULTATS MARQUANTS

C.5 TRAVAUX SPECIFIQUES AUX ENTREPRISES (LE CAS ECHEANT)

C.6 REUNIONS DU CONSORTIUM (PROJETS COLLABORATIFS)

Nous avons tenu notre quatrième réunion plénière à Paris, le 2 Juillet 2012. Tous les membres du projet ont participé.

C.7 COMMENTAIRES LIBRES

D Valorisation et impact du projet depuis le début

D.1 PUBLICATIONS ET COMMUNICATIONS

Sur la période, nous avons publié les articles suivants:

- *Programmation Web Typée* [**?**] The goal of this thesis is to contribute to make Web programming safer and more flexible than it is in the solutions prevalent today. To achieve this goal, we propose a solution based on the ML language family, which brings freedom to the programmer by its multi-paradigm aspect, while providing an important level of safety thanks to static typing. In the first part, we show that it is possible to program the browser without sticking to the style of JavaScript. Our solution is OBrowser, an OCaml virtual machine in JavaScript. The implementation supports

the whole OCaml language and its library, including the preemptive concurrency model. We additionaly present a mechanism for inter-operability between the object layers of JavaScript and OCaml, that allows to use the browser's API in a type-safe way, using OCaml objects. In the second part, we give an API for document manipulations, designed to be safer and more high-level than the browser's DOM. In particular, we aim at eliminating the implicit moves performed by the DOM to maintain the tree structure, and which limit the possibilities of static typing. First, we give fDOM, a minimal formal model similar to the DOM. then, we propose cDOM, an alternative model in which moves are replaced by copy operations. We then describe FidoML, a language based on ML equiped with document manipulation features, that are well typed thanks to the use of cDOM. throughout this part, we make a special effort to design solutions flexible enough to be used in languages other than ML. In the third and final part, we show how the work, so far presented in the context of the browser, can be applied to a multi-tier model. First, we give an overview of related multi-tier research platforms. In particular we describe the solutions they provide to a selected set of language aspects specific to Web programming. then, we conclude by giving the outline of a multi-tier language, that uses the work the first two parts to built solutions to these language aspects.

- *Séparation des couleurs dans un lambda-calcul bichrome* [**?**] Dans cet article nous introduisons un λ-calcul bichrome pour expliciter une partie de l'évaluation d'un terme en précisant la localité du calcul1. L'intérêt est alors de pouvoir définir une transformation, par β-expansion, qui regroupe les expressions de même couleur. Les propriétés de correction, de terminaison et de confluence de cette transformation sont démontrées 'a l'aide de l'assistant de preuves Coq. Cette transformation est indépendante de la sémantique de communication et de synchronisation de l'application. On s'intéresse alors aux applications utilisant deux unités de calcul comme les couples client-serveur de la programmation Web. Nous abordons le passage à un λ-calcul à plus de deux couleurs et montrons les difficultés que cela engendre.

- *How to Run your Favorite Language in Web Browser* [**?**] This paper is a concise guide for developers who want to port an existing language to Web browsers, about what to do and what not to. It is based on the various experiments that have been led in the OCaml language community. In particular, it exhibits how reusing the underlying virtual machine and bytecode associated to the language can come of great help in this task.

- *Client-server Web Applications with Ocsigen* [**?**] The Ocsigen framework offers a new way to develop sophisticated client-server Web applications. It makes it possible to write as a single program both the server and client sides of a Web application, thus simplifying a lot communications and data transfers, and avoiding code duplications. It also proposes a wide set of high level concepts to program traditional Web interactions in a very concise way while mixing them seamlessly with client side features. The use of a powerful type system improves a lot the reliability of programs, reducing debugging time, and making the code easier to maintain.

- *HipHop: A Synchronous Reactive Extension for Hop* [**?**]: HOP is a SCHEME-based language and system to build rich multi-tier web applications. We present HIPHOP, a new language layer within HOP dedicated to request and event orchestration. HIPHOP follows the synchronous reactive model of the Esterel and ReactiveC languages, originally developed for embedded systems programming. It is based on synchronous concurrency and preemption primitives, which are known to be key components for the modular design of complex temporal behaviors. Although the language is concurrent, the generated code is purely sequential and thread-free; HIPHOP is translated to HOP for the server side and to straight JavaScript for the client side. With a music playing example, we show how to modularly buid non-trivial orchestration code with HIPHOP

- *An Interpreter for Server-Side Hop* [**?**]: HOP is a Scheme-based multi-tier programming language for the Web. The client-side of a program is compiled to JavaScript, while the server-side is executed by a mix of natively compiled code and interpreted code. At the time where HOP programs were basic scripts, the performance of the server-side interpreter was not a concern; an inefficient interpreter was acceptable. As HOP expanded, HOP programs got larger and more complex. A more efficient interpreter was necessary. This new interpreter is described in this paper. It is compact, its whole implementation counting no more than 2.5 KLOC. It is more than twice faster than the old interpreter and consumes less than a third of its memory. Although it cannot compete with static or JIT native compilers, our experimental results show that it is amongst the fastest interpreters for dynamic languages.

- *HopTeX - Compiling HTML to LaTeX with CSS* [**?**]. HopTeX is a new application for authoring Html and LaTeX documents. The content of the document is either expressed in Html or in a blending of Html and a dedicated wiki syntax, for the sake of conciseness and readability. The rendering of the document is expressed by a set of Css rules. The main originality of HopTeX is to consider LaTeX as a new media type for Html and to express the compilation from Html to LaTeX by the means of dedicated style sheet rules. HopTeX can then be used to generate high quality documents for both paper printed version and electronic version. The online version of this paper is available at the HopTeX web page.

  HopTeX is implemented in Hop, a multi-tier programming language for the Web 2.0. This implementation extensively relies on two facilities generally only available on the client-side that Hop also supports on the server-side of the application: DOM manipulations and Css server-side resolutions.

- *Multitier Programming in Hop - A first step toward programming 21st-century applications* [**?**]

- *Reasoning about Web Applications: An Operational Semantics for HOP* [**?**]. We have proposed a small-step operational semantics to support reasoning about Web applications written in the multitier language HOP. This semantics covers both server side and client side computations, as well as their interactions, and includes creation of Web services, distributed client-server communications, concurrent evaluation of service requests at server side, elaboration of HTML documents, DOM operations, evaluation of script nodes in HTML documents and actions from HTML pages at client side. We also model the browser same origin policy (SOP) in the semantics. We propose a safety property by which programs do not get stuck due to a violation of the SOP and a type system to enforce it.

  Par ailleurs l'article *A multi-tier semantics for Hop* [**?**] a été édité par la revue *Higher Order and Symbolic Computation.*

References

[1] Vincent Balat, Pierre Chambart, and Grégoire Henry. Client-server Web applications with Ocsigen. In *WWW2012 dev track proceedings*, page 59, Lyon, France, April 2012.

[2] G. Berry, C. Nicolas, and M. Serrano. HipHop: A Synchronous Reactive Extension for Hop. In *Proceedings of the PLASTIC'11 workshop*, Portland, USA, October 2011.

[3] G. Boudol, Z. Luo, T. Rezk, and M. Serrano. Reasoning about Web Applications: An Operational Semantics for HOP. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 34(2), 2012.

[4] Benjamin Canou. *Programmation Web Typée.* These, Université Pierre et Marie Curie - Paris VI, October 2011.

[5] Benjamin Canou, Emmanuel Chailloux, and Jérôme Vouillon. How to Run your Favorite Language in Web Browsers. In *WWW2012 dev track proceedings*, pages –, Lyon, France, April 2012.

[6] Emmanuel Chailloux and Bernard Serpette, P. Séparation des couleurs dans un lambda-calcul bichrome. In *JFLA - Journées Francophones des Langages Applicatifs - 2012*, Carnac, France, February 2012.

[7] B. Serpette and M. Serrano. An Interpreter for Server-Side Hop. In *Proceedings of the DLS'11 symposium*, Portland, USA, October 2011.

[8] M. Serrano. HopTeX - Compiling HTML to LaTeX with CSS. In *Online Proceedings of the Scheme'11 workshop*, Portland, USA, October 2011.

[9] M. Serrano and G. Berry. Multitier Programming in Hop - a first step toward programming 21st-century applications. *Communications of the ACM*, 55(8):53–59, August 2012.

[10] M. Serrano and C. Queinnec. A multi-tier semantics for Hop. *Higher Order and Symbolic Computation*, 23(4):409–431, 2012.

D.2 AUTRES ELEMENTS DE VALORISATION

- Diffusion de logiciels:
  - Hop *2.3.0* (06/2012), `http://hop.inria.fr`
  - Js_of_ocaml *1.1* (03/2012), `http://ocsigen.org`
  - Eliom *2.0* (09/2011) et `2.1` (03/2012), `http://ocsigen.org`
  - Ocsigen Server *2.0* (09/2011), `http://ocsigen.org`
  - HopTeX *1.0.0* (10/2011), `http://hop.inria.fr`
- Création d'entreprise pour la diffusion de Ocsigen est en cours de réalisation (concours Oséo en cours, recherche de financements pour la phase de maturation en cours).
- Stand Ocsigen à la conférence WWW2012.
- Participation au barcamp IT Translation.

D.3 POLES DE COMPETITIVITE (PROJET LABELLISES)

D.4 PERSONNELS RECRUTES EN CDD (HORS STAGIAIRES)

1. **Pierre Chambart** aujourd'hui M2 France 0 PPS ingénieur janvier 2011 15 mois
2. **Grégoire Henry** aujourd'hui M2 France 0 PPS ingénieur janvier 2011 15 mois
3. **Benedikt Becker** aujourd'hui M2 Allemagne 0 PPS ingénieur novembre 2011 12 mois

D.5 ETAT FINANCIER

Non fourni pour le rapport T0+30.

E Annexes éventuelles