受付番号	0435

京都大学次世代研究者育成支援事業「白眉プロジェクト」提案書

１．応募基本情報（下記の事項について 2013 年 4 月 1 日現在で記入して下さい。本項目のページ数：2 ページ以内）

フリガナ	オカベ キワム				（顔写真） ※撮影後 3 ヶ月以内の顔写真の電子データを貼り付けてください。 （脱帽・正面・無背景） たて 40mm よこ 32mm 程度
氏　　名	岡部 究				
住　　所	神奈川県横浜市緑区十日市場町 1258 5-6-306				
連 絡 先	電　話	090-3524-7064			
	メール	kiwamu@debian.or.jp			
生年月日	1976 年 05 月 14 日	年　齢 （2013/4/1 現在）	36	性　別	男
学　　位 （取得年月日）	東京都立大学 修士 / 工学研究科電気工学専攻 電気・電子工学	国　籍	日本		
現在の所属先	ミラクル・リナックス株式会社	職　名	プリンシパルエンジニア		
希望する職種	□准教授、V 助教（必ずどちらか一方にチェックをして下さい）				
研究場所について	京都大学大学院情報学研究科通信情報システム専攻 コンピュータ工学講座計算機ソフトウェア分野 五十嵐研究室（可能な限り受入研究者名まで記載して下さい）		V 了解済み、□希望 （どちらかにチェックをして下さい。）		
関連研究分野 （必ず 2 つ選択すること）	1	分科名：計算基盤　細目名：計算機システム （平成２５年度科学研究費助成事業　系・分野・分科細目表から最も近い「細目」を一つ選択し、その「分科」とともに記載してください）			
	2	分科名：計算基盤　細目名：ソフトウェア （平成２５年度科学研究費助成事業　系・分野・分科細目表から 2 番目に近い「細目」を一つ選択し、その「分科」とともに記載してください）			

学歴及び職歴（次ページ続く）

1999 年 03 月	東京都立大学 大学卒業/工学部電気工学科 電気・電子工学 研究概要: 光ファイバの物性モデル計算
2001 年 03 月	東京都立大学 修士卒業/工学研究科電気工学専攻 電気・電子工学 研究概要: 水晶振動子のマルチモードを使用した気体センサの作成 • Ruby 製 GPIB ライブラリを使った実験 • Debian-JP http://www.debian.or.jp/ メンバーになる
2001 年 04 月	株式会社リコー入社

2001 年 07 月	社内初の x86 アーキティクチャコピー機の BIOS/bootloader 開発（主担当） mips アーキティクチャのコピー機を x86 アーキティクチャを用いて開発するプロジェクトに新人として配属。 開発したソフトウェア部品は imagio シリーズ http://www.ricoh.co.jp/imagio/ として現在も出荷され続けている。主な開発項目は以下の通り。 • 一般的な BIOS へのカスタマイズ要求仕様策定 • 機種抽象化データ構造の策定 • SD カード起動 Option BIOS の設計
2002 年 11 月	組込み機器 認証起動方式の開発 bootloader 部分 (主担当) セキュリティ強化のため認証起動方式を検討/実装した。主な開発項目は以下の通り。 • bootloader への公開鍵アルゴリズム搭載 • SD カード上への鍵情報搭載方式の検討
2003 年 10 月	海外委託先と新規 BIOS 開発 (主担当) BIOS メーカに日本法人がなかったため、海外にて要求仕様協議。前回契約での BIOS と互換の BIOS を他メーカで実現。imagio シリーズにて製品搭載。
2004 年 06 月	コピー機起動時間を 10 秒に (主担当) 30 秒程度かかっていたコピー機の起動時間を 10 秒まで短縮することに成功。起動時間を測定するログ収集/解析ツールを考案/実装。アプリケーション群を複数グループに分割。初期画面表示関連グループのみを OS 起動直後にストレージから読み込み起動。上記の測定ツールと解析手法は今も imagio シリーズの製品化毎に実施されている。
2006 年 04 月	スレッドライブラリ開発 (主担当) これまでユーザ空間スレッド(pthread 互換)で構築されていたコピー機のシステムを NetBSD-2.0 標準の m:n スレッドで再構築した。以下多くの改修を行なった。 • kernel,libc,libpthread におけるスレッドセーフ/キャンセルセーフ調査 • m:n ライブラリに不足していた API を追加。移行マニュアル作成 • シグナル周りの不具合修正。アプリ側でのシグナル取り扱いマニュアルを作成
2006 年 04 月	NetBSD-2.0 の組込みポーティング/開発 (メンバー) NetBSD-1.5→2.0 への移行に際してあらゆるサポート。
2008 年 04 月	NetBSD OS 開発/技術サポート (リーダー) 既存機種の障害解析と、新規機能開発を行なうチームの技術リーダに就任。機種群全体のプロジェクト管理リーダと二人で OS 開発を担い、実作業は完全に外部委託化。 • 仮想メモリでのページ回収コードに不具合があり、修正 patch を協議の上作成 • wifi 環境でソケットが close 不能。送信遅延によるソケット切断不能が原因 • 電源ボタンをソフトウェア検知してソフトウェアによって電源を落すしくみをリレーと watchdog のみのハードを用いてソフトウェア側で対応
2010 年 04 月	新規 x86 アーキティクチャへの OS ポーティング (リーダー) 新規 x86 ボードへの NetBSD-2.0 移植。また BIOS メーカ選定/委託締結。 • BIOS メーカを 3 者で選定。細かい評価項目を使って得点評価の後選定 • 新人時代に作成した Option BIOS を捨てて、新 BIOS 構成に仕様変更 • OS ポーティング前に起動時間を予測。起動時間改善案を提案+スケジュール
2012 年 03 月	ミラクル・リナックス株式会社入社 • PowerPC Linux の SMP 排他による不具合検証/修正 • PowerPC 版 crash コマンド(デバッグツール)の不具合修正 • NSIS を使った新しい Windows アプリケーションインストーラの作成

２．採用後に取り組む研究について

（以下の各項目について簡潔に記載下さい。本項目のページ数：5 ページ以内）

（１）取り組む研究のタイトル

・本欄には、本提案に係る研究のタイトルを日本語の場合は 30 字以内で、英語の場合は 20 ワード以内で記載して下さい。

型推論をそなえた言語による UNIX ライク OS の研究開発と実証実験

（２）研究の概要

・本欄には、本提案に係る概要を日本語の場合は 100 字程度で、英語の場合は 50 ワード程度で記載して下さい。

UNIX ライク OS の kernel は C 言語で設計されているがこれらの言語は型安全でない。本研究では左記 kernel を型推論を持つ安全な言語によって実装する。また OS の実装に必要なコンパイラも同時に開発する。

（３）研究計画等

・本欄には、研究目的、研究計画・方法・場所、研究成果、業績、その他アピールしたいことについて自由に記述して下さい。また、必要に応じて図表を入れることも可とします。

【目的】

現在 C 言語によって設計されている UNIX ライク OS の kernel を強い型を持つ C 言語より安全な言語によって再設計する。また、当該言語のコンパイラの実装を探求する。

【計画】

本研究によって作成するコンパイラの名前を「Ajhc http://ajhc.metasepi.org/」、OS (kernel 部分のみ) の名前を「Metasepi http://metasepi.org/」と呼ぶ。

1. Ajhc コンパイラの研究開発
2. Metasepi kernel の研究開発
3. Metasepi プロジェクトの啓蒙
4. Ajhc コンパイラの収益化

【方法】

1. Ajhc コンパイラの研究開発

jhc Haskell コンパイラを元にして Ajhc という名前で新しい Haskell コンパイラの研究開発をオープンソースにて行なう。jhc の実装については判明していないことが多いため、jhc コンパイラの内部実装解析/レポート作成する。Haskell コミュニティで一般的に使われているライブラリ群である Haskell Platform(参考文献 1)の内、jhc コンパイラではほんの一部のライブラリにしか対応していない。この Haksell Platform のほとんどを Ajhc に移植できないか検討/実装する。(元ライブラリの実装によっては

原理的に移植できないケースも考えられる)Ajhc のコンパイル結果の品質を向上/維持させる。そのためには多くのライブラリ/アプリケーションの移植と、継続的インテグレーションの実施が必要となる。Ajhc によってコンパイルされたコードの再入/並列実行の実現させる。実現手法について理論的な裏付けの研究も同時に行なう。GC 停止時間を低減させる。場合によっては世代別 GC の導入を検討する。OS の割り込みハンドラを実現させるため、GC の一時中断/再開の実現させる。Ajhc のコンパイル済みコードは C 言語なのでデバッグ効率は比較的良いが、 Haskell コードレベルでのデバッグ環境を充実化させる。具体的にはデバッガによる任意の Haskell サンクでの実行停止など。jhc の過去のリリースでは部分的なリージョン推論が実現されていた。その具体的な手法を調査し、現バージョンの Ajhc において部分的なリージョン推論を実現できないか検討する。 このリージョン推論のカバー率によっては GC 効率が悪くても製品実装に使えるコンパイラを作成できる可能性がある。Ajhc のランタイムは既に C 言語 3000 行のみととても小さいが、このランタイムをさらに小さくできないか検討する。 具体的には Ajhc コンパイラランタイムの一部 (GC など)を FPGA を使ってハードマクロ化できないか検討する。

2. Metasepi kernel の研究開発

ML(参考文献 2)以上の強い型を持つ言語で UNIX ライク OS の kernel を設計できないか研究する。
また、その際 UNIX ライク kernel を型によって表現する手法についても同時に探求する。
さらに UNIX ライク kernel が設計対象として大きすぎた場合にはそれより小さな FreeRTOS(参考文献 3)などの OS を強い型によって設計できないか挑戦する。
設計中の kernel の品質が劣化しないように品質維持方法の検討を行なう。

3. Metasepi プロジェクトの啓蒙

右記 3 種類の勉強会を開催する。Haskell コンパイラソースコードリーディング、スタート低レイヤー、組み込み haskell。開催場所は京都もしくは東京近郊を予定。NetBSD kernel 内部詳細理解と人材教育を目的として、NetBSD kernel マニュアルページ翻訳環境の整備する。Metasepi プロジェクト最新の研究成果を ESEC(参考文献 4)や OSC(参考文献 5)などの展示会に出展する。研究成果について論文化してプロジェクトの意義を啓蒙する。

4. 収益化

Ajhc コンパイラの収益モデルの確立と Ajhc 使用者を増加を目的として以下のようなアプリケーション例を実現する。 Metasepi 本体の収益化は白眉期間の 5 年間では達成できない見込み。

a. マイコンをターゲットとした Ajhc 統合開発環境

狙い: マイコンチップの製造メーカとの協業
実施例: 組込み開発学習教材 EAPL-Trainer (参考文献 6)
実施例の課題: 動的型付け言語を使っているため実行時エラーが起きる可能性が高い

b. マイコンを使ったアプリケーションの作成 (例えば模型飛行機制御など)

狙い: Ajhc の具体的な応用例を示すことでコンパイラの利用者数向上
実施例: Galois 社による GHC Haskell コンパイラと DSL を使った模型飛行機制御実験 (参考文献 7)

実施例の課題: DSL を使った設計をしているため Ajhc を用いて汎用言語での設計を行なった方が設計効率が良い

c. クライアント PC からソースコードを Ajhc を内包したサーバに送ってコンパイル (サーバサイドコンパイラ)

狙い: Ajhc コンパイラの導入コスト低減/サーバサイドで Ajhc の不具合検出をすることによる Ajhc の品質向上

実施例: ARM 社 mbed プラットフォーム (参考文献 8)

実施例の課題: 設計に用いる言語が C 言語であるため表現力が乏しく型が弱いため実行時エラーの可能性が高い。デバッガを使うことができない。

d. Haskell と FPGA によるハードソフト協調設計

狙い: 小規模組込機器メーカーへのコンサルティング

実施例: IBM 社 Liquid Metal (参考文献 9)

実施例の課題: オープンソースプロジェクトではない。Java は Haskell よりも型の表現力が弱い。

【場所】

京都大学大学院情報学研究科通信情報システム専攻 コンピュータ工学講座計算機ソフトウェア分野
五十嵐研究室
(研究開発の状況によっては専門知識を持つ機関への出向する可能性有り)

【既存研究とその問題点】

ML や Haskell のような強い型を使った kernel の設計事例はいくつか現存している。(参考文献 10,11,12,13) しかしこれらのプロジェクトでは Firefox など実際のデスクトップで使用されるアプリケーションは動作しない。また、ほとんどのプロジェクトは 32bit x86 CPU のみに対応しており移植性に乏しい。このような不幸な現状の理由はこれらのプロジェクトが既存の UNIX ライク OS の実装を参考にせず、独自に kernel の設計を行なっているためだと考えられる。実際に上記プロジェクトのソースコードを調査するかぎりでは x86 のみに対応しており、デバイスドライバもバスドライバのなどの抽象化をなされていない。これらのプロジェクトには日々デスクトップとして使える OS を実現する見込みがほとんどないと考えられる。

【現状の成果】

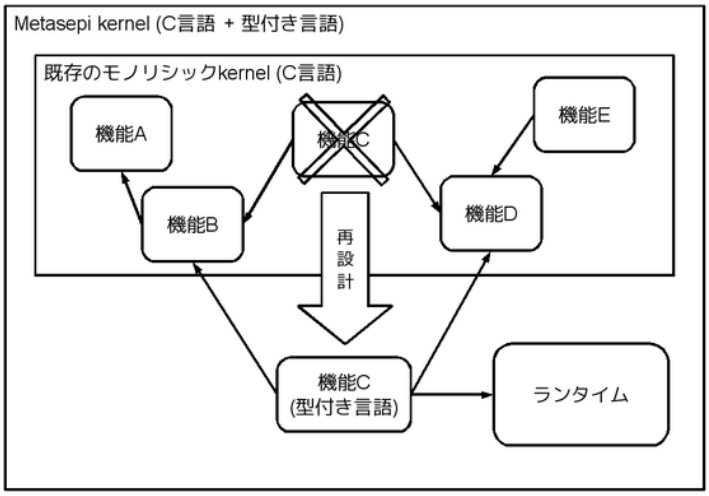
1. Ajhc コンパイラの研究開発

オープンソースの Haskell コンパイラ jhc(参考文献 14)を元にして Ajhc コンパイラ(参考文献 15)の開発を開始。2013 年 05 月 05 日現在、Ajhc を 2 回正式リリース済み。(参考文献 16,17) Windows 上での Ajhc コンパイラの実行をサポート済み。これらの修正の一部は元にした jhc コンパイラに取り込まれている。英語の jhc ユーザーズマニュアル(参考文献 18)を日本語訳(参考文献 19)して公開。Ajhc の内部実装について解析中。解析の途中経過を Metasepi プロジェクト Blog にて公開。(参考文献 20,21) Ajhc コンパイラランタイムの一部 FPGA 化について検討開始。ソフトマクロ CPU が必要になることから openrisc(参

考文献 22)に修正を加える案が有力。Ajhc コンパイル結果の品質向上のために継続的インテグレーション環境を導入。(参考文献 23) 上記環境にて Ajhc ソースコード修正毎にビルド+回帰テストが自動実行される。この回帰テストにてランタイムエラーが 2 件発生している。1 件は任意精度整数が全うな実装でないため 0 除算がランタイムに発生している。対策検討中。Ajhc でコンパイルされたバイナリの解析のため 、Haskell ヒープの内部構造をダンプするツールを作成。(参考文献 24)

2. Metasepi kernel の研究開発

型による UNIX ライク kernel を設計するにあたって下図のように既存の C 言語で書かれた kernel 実装を少しずつ強い型を持つ言語で再設計することにした。この設計手法を採用することによってコンパイル可能/動作可能な状態を維持しながら研究開発を行なうことが可能となる。この設計手法が少なくとも割り込み禁止のソフトウェアには有効であることを NetBSD bootloader に適用することで実証済み。(参考文献 25) この手法は動作可能な状態を維持しながら型による再設計の度にテストを実行できるため、Metasepi kernel の品質維持にも有効である。



3. Metasepi プロジェクトの啓蒙

Haskell コンパイラ内部実装の勉強会を 2 回開催済み。(参考文献 26,27)
NetBSD kernel 内部実装の勉強会を 1 回開催済み。(参考文献 28)
NetBSD kernel マニュアルページ翻訳環境の整備を開始。現在翻訳環境の再設計中。(参考文献 29)

4. 収益化

マイコンをターゲットとした Ajhc 統合開発環境の手始めとして、Cortex-M3 マイコンでの Haskell コード実行を実現。
デモソースコード公開。デモのサポートプラットフォームを拡大中。(参考文献 30)

[参考文献]

1. The Haskell Platform http://www.haskell.org/platform/
2. ML (programming language) http://en.wikipedia.org/wiki/ML_(programming_language)
3. FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems supporting 33 microcontroller architectures http://www.freertos.org/

4. 組込みシステム開発技術展 (ESEC) http://www.esec.jp/
5. OSPN - We Are Open!! http://www.ospn.jp/
6. 組込み開発学習教材 EAPL-Trainer（イープル トレーナー）mruby : ILC「株式会社アイ・エル・シー」 http://www.ilc.co.jp/commodity/eapl-trainer-mruby/index.html
7. Galois - Blog - Copilot: a DSL for Monitoring Embedded Systems http://corp.galois.com/blog/2010/9/22/copilot-a-dsl-for-monitoring-embedded-systems.html
8. mbed http://mbed.org/blog/entry/mbed-enabled-Freescale-FRDM-KL25Z-board/
9. Liquid Metal http://researcher.watson.ibm.com/researcher/view_project.php?id=122
10. Funk (OCaml 製) http://home.gna.org/funk/
11. snowflake-os (OCaml 製) http://code.google.com/p/snowflake-os/
12. House (Haskell 製) http://programatica.cs.pdx.edu/House/
13. HaLVM (Haskell 製) http://corp.galois.com/halvm/
14. jhc http://repetae.net/computer/jhc/
15. Ajhc - arafura-jhc http://ajhc.metasepi.org/
16. [jhc] ANNOUNCE: Ajhc 0.8.0.2 Release http://www.haskell.org/pipermail/jhc/2013-March/001028.html
17. [jhc] ANNOUNCE: Ajhc 0.8.0.3 Release http://www.haskell.org/pipermail/jhc/2013-April/001047.html
18. Ajhc User's Manual http://ajhc.metasepi.org/manual.html
19. Ajhc ユーザーズマニュアル http://ajhc.metasepi.org/manual_ja.html
20. jhc コンパイルパイプラインの全体像 - Metasepi http://metasepi.org/posts/2013-01-31-jhc_internal_overview.html
21. jhc コンパイルパイプライン : Grin => C - Metasepi http://metasepi.org/posts/2013-02-14-jhc_grin_to_c.html
22. Getting Started with OpenRISC http://kevinmehall.net/openrisc/guide/
23. Travis CI - Free Hosted Continuous Integration Platform for the Open Source Community https://travis-ci.org/ajhc/ajhc
24. ajhc/utils/gdb_jhc.py at arafura · ajhc/ajhc · GitHub https://github.com/ajhc/ajhc/blob/arafura/utils/gdb_jhc.py
25. デザイン Arafura - Metasepi http://metasepi.org/posts/2013-01-09-design_arafura.html
26. HaskellJP wiki - Workshop/ReadGHC/0 http://wiki.haskell.jp/Workshop/ReadGHC/0
27. HaskellJP wiki - Workshop/ReadGHC/1 http://wiki.haskell.jp/Workshop/ReadGHC/1
28. Meeting0 · start-printf/wiki Wiki https://github.com/start-printf/wiki/wiki/Meeting0
29. netbsdman.masterq.net http://netbsdman.masterq.net/
30. ajhc/demo-cortex-m3 · GitHub https://github.com/ajhc/demo-cortex-m3

３．研究業績について

・本項目のページ数：4 ページ以内

・本欄には、これまでに発表した論文、著書、産業財産権、招待講演等のうち、主要なものを選定し、各区分（論文、著書、産業財産権等の区分）毎に現在から順に発表年次を過去にさかのぼり、通し番号を付して記入してください。なお、学術誌へ投稿中の論文を記入する場合は、掲載が決定しているものに限ります。

・例えば発表論文の場合、論文名、著者名、掲載誌名、査読の有無、巻、最初と最後の頁、発表年（西暦）について記入してください。以上の各項目が記載されていれば、項目の順序を入れ替えても可とします。

・著者名が多数にわたる場合は、主な著者を数名記入し以下を省略（省略する場合、その員数と、掲載されている順番を○番目と記入）しても可とします。なお、応募者には下線を付して下さい。

[著書]

1. 「Lighter than Light」 岡部究. λカ娘 4 巻. 査読無し. 15-34 頁. 2012 年 12 月
2. 「RTS 海溝二万マイル」 岡部究. λカ娘 3 巻. 査読無し. 33-54 頁. 2012 年 08 月
3. 「Copilot への希望と絶望の相転移」 岡部究. λカ娘 2 巻. 査読無し. 7-21 頁. 2011 年 12 月
4. 「OS を侵略しなイカ？」 岡部究. λカ娘 1 巻. 査読無し. 35-39 頁. 2011 年 08 月
5. 「 Multimode Quartz Crystal Microbalance 」 GOKA Shigeyoshi, OKABE Kiwamu, WATANABE Yasuaki, and SEKIMOTO Hitoshi. Japanese journal of applied physics. 査読有り. 3073-3075 頁. 2000 年 05 月

[講演]

6. 「Haskell ではじめる Cortex-M3 組込みプログラミング」 オープンソースカンファレンス 2013 Tokyo/Spring / ライトニングトーク@Business Day. 2013 年 02 月
7. 「スタート低レイヤー #0」オープンソースカンファレンス 2013 Tokyo/Spring / NetBSD のご紹介. 2013 年 02 月
8. 「What is Metasepi?」 秘密結社 Metasepi 作戦会議. 2013 年 02 月
9. 「Dive into RTS - another side」GHC ソースコードリーディング勉強会 第 1 回. 2012 年 09 月
10. 「Emacs と Gloss でお絵描きしてみるよ」Haskell Day 2012. 2012 年 05 月
11. 「NetBSD man を翻訳しよう! (OSC2012 版)」 オープンソースカンファレンス 2012 Tokyo/Spring . 2012 年 03 月
12. 「GHC ソースコード読みのススメ」 GHC ソースコードリーディング勉強会 第 0 回. 2012 年 02 月
13. 「Debian Loves Haskell」Tokyo Linux Users Group (TLUG) Technical Meeting. 2011 年 11 月
14. 「Haskell と Debian の辛くて甘い関係」 第 81 回東京エリア Debian 勉強会. 2011 年 10 月
15. 「Cairo でほっかり GUI プログラミング」 第 0 回 スタート Haskell. 2011 年 07 月
16. 「Wiki 設置するなら gitit!」 第 6 回 qpstudy06. 2011 年 05 月

[オープンソース活動]

17. Ajhc Haskell コンパイラの開発 http://ajhc.metasepi.org/ 2013 年 03 月〜
18. Metasepi OS の研究開発 http://metasepi.org/ 2012 年 08 月〜
19. プレゼンテーションツール Carettah の開発 http://carettah.masterq.net/ 2011 年 07 月〜
20. NetBSD man ページ翻訳環境の整備 http://netbsdman.masterq.net/ 2011 年 04 月〜
21. Debian Maintainer による複数の Debian パッケージの管理 2009 年 09 月〜

４．応募の動機と白眉研究者としての抱負について

（以下の各事項について簡潔に記載下さい。なお、本項目については主として第２次審査の面接において使用します。本項目のページ数は２ページ以内とします。なお２ページの範囲内であれば、必要に応じて各回答欄の大きさを調整いただいて構いません。）

（１）白眉プロジェクトへ応募した理由を記してください。

・これまでのあなたのキャリアと白眉プロジェクト終了以後のあなたの思い描くキャリアに触れつつ、あなたが本公募に応募した理由を記載してください。

私は株式会社リコーにて 10 年間大規模組み込み開発を経験してきました。その開発を通じて機能開発はもちろんですが、それよりも多くの不具合/市場障害を経験しました。私はこの不具合とそれによる開発者の不幸の根本原因がいったい何なのか、この数年間追い求めてきました。そして型推論をそなえた言語について学んだ結果 「根本的な原因は型推論のない弱い型を使ってソフトウェアを設計していることにある」と確信するようにりました。さらに「OS のような最新の注意をはらって設計しなければならない基盤ソフトウェアがなぜ弱い型を使って設計されているのか」疑問を持つようになりました。

この問題がコンピュータアーキティクチャ全体にとって明確に重要であるにもかかわらず、ほとんどの開発者は注意を向けていません。もしくは実験的な OS を作ってその後放置されています。その理由は単純です。既存 OS が十分なクオリティを確保しており、また既存 OS の上で動くプログラムを作る方が OS を作るより圧倒的に簡単だからです。もちろん既存 OS と同等の機能を持つ OS を型推論を持つ言語によって設計することは大変困難です。この既存 OS クオリティがどのように担保されているかについても人々は盲目です。マイクロソフトはどのようにして Windows のクオリティを維持しているのでしょうか？ Linux kernel のようなクオリティはオープンソース開発プロセスを使うだけで維持できるのでしょうか？そのノウハウは工学的な解決をなされているのでしょうか？さらにこの問題を解決することに研究者は積極的ではありません。たしかに彼らが作る OS は確かに新規性にあふれています。しかし実用からはかけはなれています。彼らの作った OS の上で FIrefox や Emacs は動きません。日常用途として使わないソフトウェアはいずれ朽ちる運命にあります。

私はこの問題を解決するために Metasepi http://metasepi.org/ という UNIX ライク OS を型推論をそなえた言語によって開発するプロジェクトを開始し、まずは OS を記述するための言語処理系である Ajhc コンパイラ http://ajhc.metasepi.org/ の開発に着手しました。これらのプロジェクトを通して私が生きている内にこの問題に対する解を得たいと強く思っています。

これらのプロジェクトはこれまで私の個人的な時間を使って行なってきました。しかし 2 つの問題に現在直面しています。(a) フルタイムでの開発でないために開発工数が不足している。 (b) Haskell コンパイラの開発にはアカデミックな知識と議論が不可欠だが近くに研究者がいないためアカデミックからの議論が不足している。 収益化の見込が立てば弊社にてフルタイムで Ajhc コンパイラの開発が可能ですが、それだけでは(b)の問題を解決できません。そのため今回、白眉プロジェクトに応募しました。

（２）あなたにとっての理想の研究者像とはどのようなものですか？

・あなたにとっての理想の研究者像を簡潔に記載して下さい。

目的のためには手段を選ばない技術者を理想としています。また学術的な通説(もしくは権威ある意見)に

ついても検証が終わるまでは中立の立場を貫くようにしようと考えています。
具体的には、Metasepi OS を開発するにあたり既存のコンパイラを物色しましたが Metasepi の実装に必要な機能を持つコンパイラは見つかりませんでした。Haskell コンパイラとしてスタンダードである GHC が組み込み開発には絶望的に使えないと判明した後、jhc という世間からはほとんど注目されていないコンパイラが組み込み向けに適しているということを jhc のソースコード解析から調べ、組み込み用途向けマイコンでのデモを作りました。また jhc に関する論文は一件もありません。そのため研究者達はこの素晴しいコンパイラに対する理解を放棄してしまっています。
これからの Metasepi と Ajhc の開発においても学術的な通説を疑い、ありとあらゆる手段をつくすつもりです。例えば、ランタイムを持つ言語で OS を設計する際には通常の UNIX ライク kernel のように「割り込みハンドラ」を使うのではなく、「割り込み要因をポーリング」した方が良いとされています。これについても中立の立場を取る予定です。現状、前者による実装が困難なのであれば、前者を容易に実現可能なコンパイラを作れば良いのです。 さらに現状では Ajhc のランタイムとコンパイルパイプラインはシンプルになっていますが、今後肥大化/複雑化する可能性があります。このような状況を制御するために形式手法が必要であれば、 当該分野を学習します。
私は自分の専門領域の中にのみ注力することが目標達成のための最善ではないと考ます。専門外にある分野を自分で理解し、自分の中にある解を発見しようと努めます。専門家の意見は尊重します。しかし解は他者である専門家の中にあるのではなく、ステークホルダーである自分の中にしかないと信じています。

（３）一人の研究者として、現在、世界が抱える諸課題の解決に向けてどのような貢献が出来るとお考えですか？

・どのような課題でも結構ですので、簡潔に記載下さい。

コンピュータアーキティクチャを内包する世界中の機器に対して安全な設計手法を提供します。この設計手法を使うことによって (a) 組込製品開発における設計工数を今より予測可能なものにします。 (b) 組込製品開発における改造工数を削減します。 (c) 古い UNIX ライクインターフェイス(POSIX)よりも柔軟で安全な OS/アプリケーション間インターフェイスを実験する下地ができます。
これら 3 つの利点の恩恵は組み込み機器だけにとどまりません。現在多くの機器で使われている POSIX インターフェイスは 20 世紀に発明されました。Windows 以外の世界中のソフトウェアアーキティクチャはほとんどが POSIX インターフェイスを元にしています。Metasepi を開発することは組込機器のみならず世界中すべてのコンピュータアーキティクチャを安全な設計へと変革することに繋がります。
さらに遠い未来ですが、最終的には Linux 配付版の一つである Debian http://debian.org/ の移植版として Metasepi を提供することを考えています。既存の Linux OS ユーザには今迄と同じ使い勝手の提供と、型推論を持つ言語による安全な設計を同時に満たすことが可能になります。これは既存のソフトウェア群を安全な設計へと移行させる橋渡しとなるはずです。つまり豊富な既存資産と型推論による安全な設計とを協調動作させることができます。研究者はこの Debian GNU/Metasepi プラットフォームの上でさまざまなソフトウェア設計手法を実験することができ、また組み込み機器メーカーは左記に改造を加えて製品開発行なうため現在の設計手法よりもはるかに安全になるでしょう。