



OSSの品質についてのアプローチ2018

2018年11月14日

OSS活用WG/技術本部本部長

竹岡尚三 (株)アックス

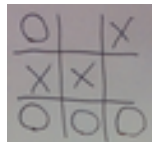
竹岡って誰?



- 1977年(高校2年生): MITのJames Slagleの書いたAIの教科書(日本語版)を読んで
「○×」を打つプログラムを8bitマイコンTK-80(8080, RAM 512Bytes)の機械語で書く
- 1980年代: UNIXのサーバクライアント方式の仮名漢字変換システムWnn、国産シンクライアント(X Window端末"XMiNT")などの設計開発に携わる。TCP/IPスタックの独自実装。
時相論理推論型言語Temporal Prologの合理的な実装などを行う
- 1990年代初頭: 1024PE規模の超並列計算機"SM-1"(豊橋技科大 湯浅研, 住友金属)のLSIハードウェア、ソフトウェアの設計開発に携わる。Common Lispで開発ツール書いた。
当時、超並列計算機は、ニューラルネットのシミュレーションによく使用された
- 1992年: (株)アックスを興し、マイクロカーネルXTALを設計。XTALはオリンパス社製デジタルカメラ、シャープ ザウルスなどに使用された。現在、アックスは、人工知能を開発&販売中
名古屋大/東大の自動運転プロジェクト"Autoware"のお手伝い
JASA技術本部長/理事, OSSコンソーシアム副会長, PCクラスタコンソーシアム理事, DEOS協会事務局長/理事



カラーザウルス(1996年)



シンクライアントXMiNT
(1988年)



超並列計算機SM-1
(1992年)



オリンパス デジカメ
(2000年)



たけおかの推論AIを高速化する特許



「ルールセットを選択可能な推論エンジンを有するプログラム、装置及び方法」

ごまめ

- 機械学習を、論理推論 機構の中心に使う
- 推論の探索空間を小さくする
→ 推論が高速に





今、OSS (オープンソース・ソフトウェア) なのか？

自動運転もOSS!「Autoware」



- ・東京大学/名古屋大学 加藤真平先生 日本で最も進んだ自動運転 研究
 - ・東大&名古屋大学 開発 自動運転ソフトウェア「Autoware」
 - ・名古屋大学ら、開発済み自動運転システム一式をオープンソース化…
加藤准教授「時間をジャンプ」
 - ・Autowareは、オープンソース・ソフトウェアとして無償配布されている
- <http://response.jp/article/2015/08/26/258648.html>



自動運転OSS「Autoware」



自動運転OSS「Autoware」



- # 「ドライバーなし自動運転」初の公道実験
- 2017/12/14 愛知県にて
 - 公道での遠隔型自動運転実証実験



なぜ自動運転か? = シェアリング・エコノミー時代



■ シェアリング・エコノミー時代

- モノを所有することの価値/魅力が減少中
- カー・シェアリングが普通になる(だろう)

■ 近距離の移動は、電動 低速モビリティ

- シェアリングされた

■ クルマを保有しないので、免許を持つ人が減る

→ 自動運転

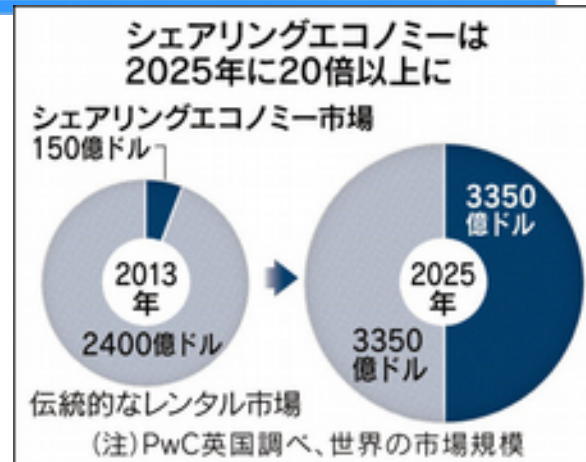
■ 自動運転で適切な配車

■ 都会のラスト・ワンマイル

- 鉄道などからの、

ラスト・ワンマイルを

自動運転車/カーシェアリン



<http://www.nikkei.com/article/DGKKZO91290540T00C15A9E8000/> より引用

シェアリングシティが解決する課題と経済成長領域





■ カーシェアリング

- 配車システムが必要

■ 自動運転+配車システム



カーシェアリング

- 無人配車
 - 要求のある場所へ、無人で配車
- 無人帰還
 - どこでも乗り捨て、無人で自動帰還

ロボット用ミドルウェア



RTミドルウェア

- 産総研などが開発しているロボット用ミドルウェア
 - ・ RTコンポーネントは、OMGにて、国際標準化
- 「OpenRTM-aist」は、RTミドルウェアの産総研による実現
 - ・ ライセンスは、LGPLおよび



産総研と個別に契約するライセンスのデュアルライセンス方式

ROS

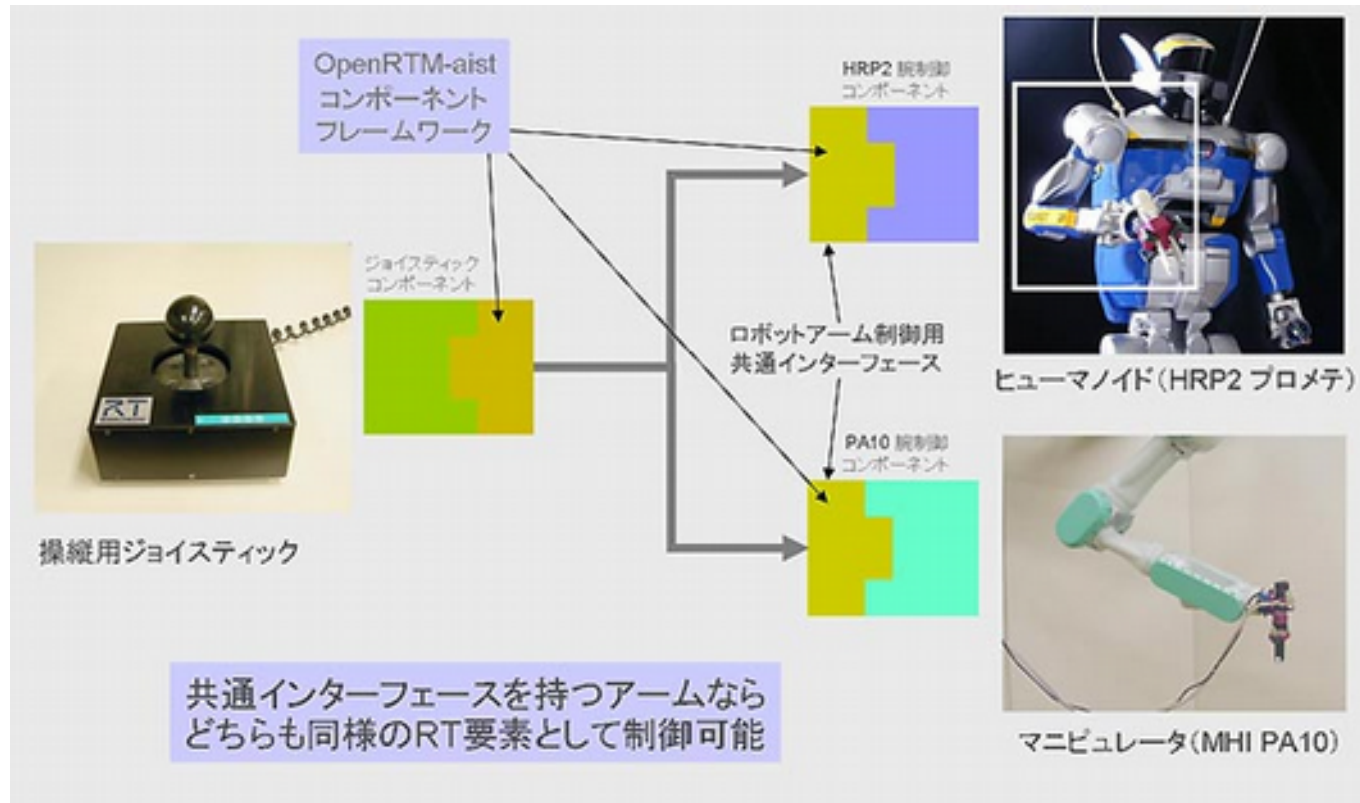
画像引用元, <http://www.openrtm.org/openrtm/ja/node/161#toc5>

参考 <http://www.openrtm.org/openrtm/ja/content/openrtm-aist-official-website>

- データの流れに応じて、コンポーネントをつなぐ
- 自動車の自動運転でも採用
- OpenCVも含まれている
- 雑に言ってしまうと…
 - ・ ロボットを作るためのソフトウェア部品の多くが含まれている



ロボット用ミドルウェアとは(RTミドルウェア)



引用元

http://www.aist.go.jp/aist_j/press_release/pr2005/pr20050224/pr20050224.html

Dronecode



<https://www.dronecode.org/>

■ Linux Foundationの取りまとめで

■ 無人飛行体 Unmanned Aerial Vehicles (UAVs)を開発
オープンソース・ソフトウェア + オープンソース・ハードウェア

■ オン・ボード(機体上)と、オフ・ボードのソフトウェアがある

- オンボード: 機体の制御、センサー制御
- オフボード: リモコン、データ収集/記録

■ Dronecode は、いくつかのプロジェクトとリポジトリでできている。

• ソースコードは、下記から

PX4 <https://github.com/PX4>

Pixhawk <https://github.com/Pixhawk>

MAVLink <https://github.com/mavlink>

UAVCAN <https://github.com/uavcan>

ROS <https://github.com/ros>

Other repositories <https://github.com/Dronecode>

■ ArdupilotはDronecodeから独立

- <https://github.com/diydrones/ardupilot>
- ライセンス・ポリシが合わなかった

画像引用元

https://www.dronecode.org/sites/dronecode/files/styles/dronecode_header/public/front_page_slides/images/drone_video_slide.png?itok=7C7IFYIP



- Arduino
 - オープンソース・ソフトウェア (OSS)
+
 - オープンソース・ハードウェア (OSHW)
 - OSS
 - 開発環境 IDE
 - ライブラリ: 膨大な量のライブラリ
 - OSSで、世界中のみんなが開発
 - OSHW
 - 回路図
 - 基板レイアウト
 - お手軽試作にバッチリ
 - ハードウェアは、アマチュア品質だが...
- Raspberry Pi
 - Linux搭載、ハードウェア情報公開



- 「無料だから嬉しい」とか言っていると、負ける
- 特定企業のOSとは違い、ソースがあるので、
理解し、独自の改良が可能
- デファクト・スタンダードがOSSなら、
特定ベンダに囲い込まれない
- 自分の都合でシステムをリリースできる
 - ・特定のソフトウェアのリリース(バージョンアップ)に
引っ張られない
 - ・独自に品質を上げたソフトウェアを、自由に維持できる

OSSに乗るしかない!



- 顧客が、OSS前提で仕様を決めている
- サービス主導の時代→OSSで素早くシステム構築
 - Windows+ブラウザで、できていることは、できて欲しい
- 高度なロボットを、すぐに作りたい
- 人工知能(AI)が入った機器をすぐに作りたい

- OSSを活用する世界をみんなで考えよう
→ JASA OSS活用WG



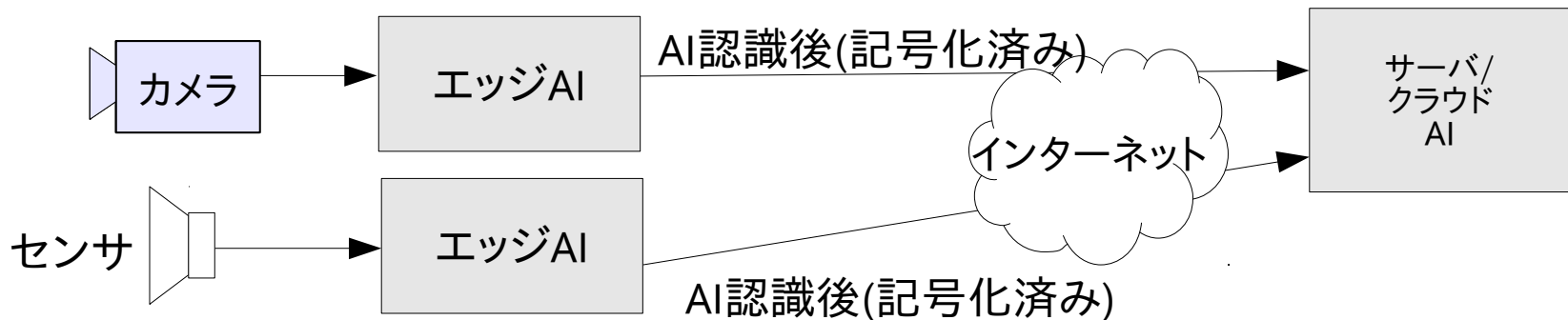
機械学習(AI) OSS

エッジAI

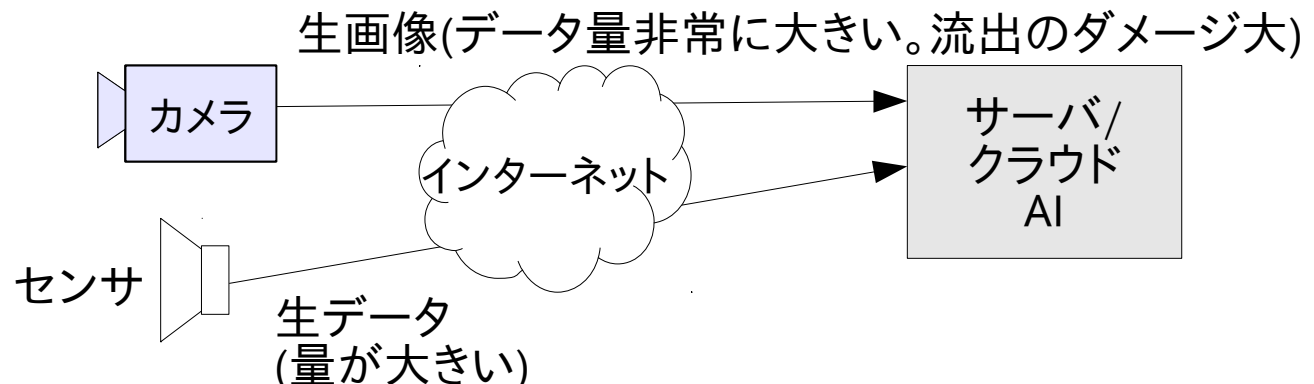


- ・センサの側で動作するAI (Raspberry/スマホ・クラスで、高速に動作する AI)
- ・生の画像を、外部へ出さない
 - ・個人の部屋の動画、プライベートな画像を、外部に出さない
 - ・他者は、サーバ上のAI に送出 → 画像/動画 流出の危険がある
- ・サーバ/クラウドへの、データの送信量を、極端に減らすことができる

エッジAI



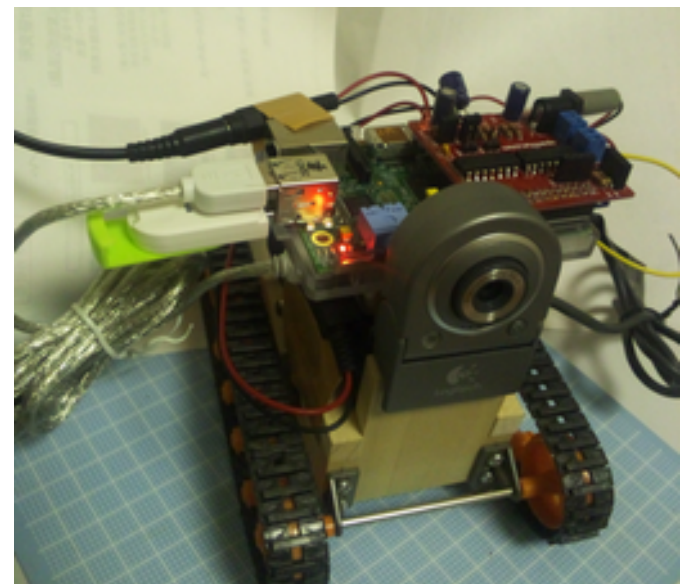
サーバ頼り



OSS機械学習ロボット



- 人間の顔を見つけたら、追尾する
 - ・ 顔の位置を判断して、右へ行くか、左へ行くかを定める
 - ・ 阪急電車も判別可能(機械学習によって)
- Linux+OpenCV
- OpenCVは
 - ・ 機械学習(SVM)による顔認識などを含む
 - ・ ROSにも含まれている
- 総合的な画像処理ライブラリ
- ハードウェア
 - ・ RaspberryPi
 - ARM11@700MHz
 - 512MBytesRAM
 - ・ USBカメラ
 - ・ DCモータ



OSS機械学習 顔追尾 ロボット



スタンドアロン。電源のみ供給

- Google, Intel などが、推論(inference)のみ実行用のチップなどを開発/出荷
- Intel Movidius Neural Compute Stick
 - プロセッサ Movidius VPU
 - サポートフレームワーク TensorFlow、Caffe
 - 接続規格 USB 3.0 Type-A
 - 消費電力も低く、割と実用的
 - RaspberryPiに接続しても使用可能
- Google TUP 第一世代は、推論むけ





人工知能(AI)

機械学習AI

ニューラルネットワーク
(1950年代末期)

パーセプトロン
(1958年)

DeepLearning
(2010年ごろ～
大流行中)

Support
Vector
Machine
(2000年ごろ～
2010年ごろ流行)

論理推論

第一次,第二次ブームの中核
(1960年 中～末期)
(1980年中期～90年代初頭)

演繹推論
(第2次AIブーム
の中核)

帰納推論
(第2次AIブームの中核
第五世代計算機
(ICOT)の中核)

確率システム
ベイズ推定

機械学習/DL は、カテゴリ分け

- ・機械学習/DLが得意
- ・判定/判断ができれば、知的でしょ
- ・知的な仕事の

大部分(?)は「判断」だ

・例

- これを買うと儲かるか?
- これを食べると健康になるか?

- ・知能は大目標に向かって
小さな判断を積み上げている



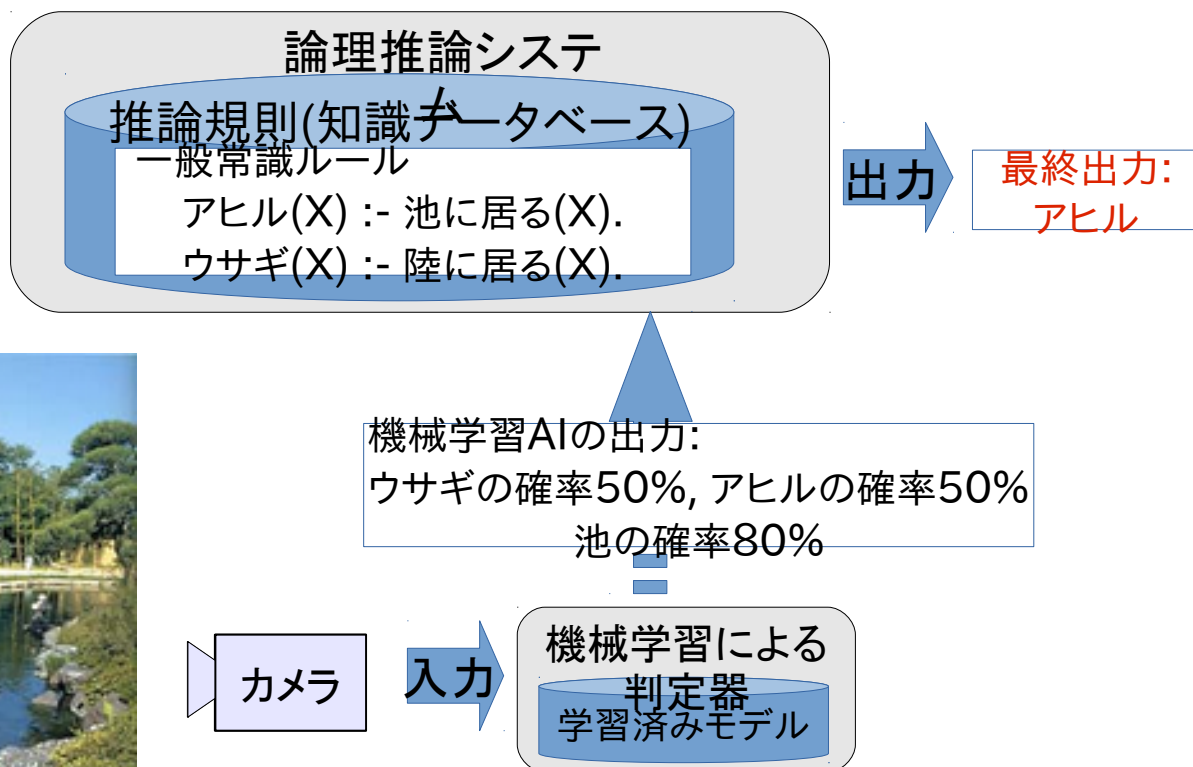
?





機械学習AIと論理推論AIのハイブリッドAI

- ・「池に居る」という文脈/背景を、論理的に使用
- ・裸のDeepLearning(機械学習AI)を、論理推論AIで補う
- ・ウサギかアヒルか、はっきり判らない画像を、正しく認識できる
- ・“常識”をルール化し、論理推論AIで常識を適用する



機械学習はDeep Learning だけじゃない!



■ 深層学習 (Deep Learning)

- 多層パーセプトロン(Perceptron)の強化形
 - パーセプトロンは、1950年代末期からある
 - バック・プロパゲーションを色んな層に
- 良い学習のためには、演算が多い
- 学習結果、判定結果は解析不能
 - 本質的に。

■ サポート・ベクタ・マシン (Support Vector Machine)

- 試料をベクトル化して、比較
 - ベクトルの要素は、機械が決定する。が、人間がベクトルを見て、解析することは可能
- Deep Learningと比べて演算が少ない

■ Google自動運転でDeep Learningが有名になるまでの10年間ほどは、SVMがとても多く使われていた

組み込みで使える Deep Learning(深層学習) OSS



- 組み込みで使えるものを紹介
 - 独立して動作
 - クラウドとかサーバは不要
 - 割と新しいもの
 - でも、学習時には、GPUが必要かも…
- TensorFlow (Deep Learning OSS)
 - Googleが開発
 - Apache 2.0ライセンス
 - AndroidやiOSでも動作
 - Pythonから使いやすい
 - <https://www.tensorflow.org/>
- Chainer (Deep Learning OSS)
 - 日本 Preferred Infrastructure 社が開発
 - ライセンス
 - 基本的に、改変、再配布など可能。詳細は下記
 - <https://github.com/pfnet/chainer/blob/master/LICENSE>
 - Pythonから使いやすい
 - X86 CPU以外で動作している情報が無い…
 - <http://chainer.org/>
- Caffe (Deep Learning OSS)
 - UC BerkeleyのBerkeley Vision and Learning Center (BVLC)が開発
 - 速い
 - Pythonから使える
 - <http://caffe.berkeleyvision.org/>

Support Vector Machine(SVM) OSS



- SVMは、計算が少ない
- ARMなどの組み込みCPUで十分に動作可能
- SVM-Light
 - Thorsten Joachims @Cornell University が開発
 - Linear kernelの学習時間は短い
 - 精度はいい
 - http://www.cs.cornell.edu/People/tj/svm_light/index.html
- SVM-perf
 - Thorsten Joachims @Cornell University が開発
 - 学習時間は短い
 - 精度はいまいち
 - http://www.cs.cornell.edu/People/tj/svm_light/svm_perf.html
- LIBSVM
 - 国立台湾大学のChih-Chung Chang と Chih-Jen Linが開発
 - 学習時間は長い
 - Linear kernelの精度は SVM-Light並
 - <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>
- LIBLINEAR
 - 国立台湾大学 Machine Learning Group が開発
 - 学習時間がかなり短い
 - 精度はいい
 - <http://www.csie.ntu.edu.tw/~cjlin/liblinear/>
- CvSVM (OpenCV)
 - OpenCVに入っている
 - よく使われている
 - http://opencv.jp/opencv-2.1/cpp/support_vector_machines.html



OSS品質評価 の アプローチ

OSSって面倒くさい？



▪品質問題

- 適当に作られた無料のソフトウェアの品質は？
- どこの誰だか、知らない人が作ったものでしょ…
- OSSの品質が、とても良い、ということは、しばしばある

▪品質の押さえ方は、大事

▪OSSの品質評価の問題点

- 開発手法で押さえられない
 - (導入したい)OSSは開発が終了しているから
- コードレビューにそぐわない
- 巨大OSSだから、使用したい
 - 小さいソフトウェアなら、新たに書き下ろせる
- 巨大OSSのコードをすべて精査するのか???
- 通常のサイトには、その能力がない

OSSの品質評価の解決方法



- テストする
- 他者の商品ソフトウェアを購入した場合
 - 製作者である他者が、品質を保証
 - テスト結果で、品質を示すことも多い
 - 必ずしもそうではないが…
- OSSもテストして、品質を確認

OSSのテスト方法の難点



- OSSもテストして、品質を確認したい…
 - OSSのテストの難点
 - 仕様がいまひとつ明確でない
 - どこを重点的にテストすればいいか判らない
 - コードをレビューしていないから、
難しそう&重要そうな部分が判らない
- ↓
- 境界条件テストが難しい
 - 限界値テストは、可能だろう

OSSのひとつのテスト方法



- 境界条件テストが難しい
 - 限界値テストは、可能だろう
 - 仕様が不明確な場合もあるが…
- ↓
- いろんな引数を、対象OSSに食わせる
「Fuzzテスト」
 - 異常な振る舞いをしたら、危険
 - 危険な値が、入力されないように、外で処置
 - or
 - OSSを修正

Fuzzテスト:OSSのひとつのテスト方法



- いろんな引数を、対象OSSに食わせる
- Fuzzテスト
 - ファズ(英:fuzz)(予測不可能な入力データ)を与えることで意図的に例外を発生させ、その例外の挙動を確認するという方法を用いる。ファズテストと呼ばれることもある。

<https://ja.wikipedia.org/wiki/>

%E3%83%95%E3%82%A1%E3%82%B8%E3%83%B3%E3%82%B0

より引用

Fuzzテスト試行により



- 品質評価の基準を探る
 - Fuzzテストの結果がどの程度だと、安心だと言えるのか
 - or
 - Fuzzテストの結果からは、安心だと言えないのか
- テスト・コストを探る
 - どれぐらいテストをすると
 - 問題が発現するのか
 - 十分にテストした、と言えるのか/言えないのか?

・下記の指標が作れるか？

1.fuzz結果から、デバグ・コストが読める

・fuzz 結果(良さ) vs 精査(デバグ)コスト の関係が掴み出せるか？

2.一般的な 生産性に似せた品質 評価指標

- ・ エンタープライズではよく統計結果が出ている
 - ・ ex. バグ/行数(ソフトウェアの規模) ← 普通の生産性指標
- ・ fuzzテスト結果から、
 - ・ バグ/行数(ソフトウェアの規模) → 新しい評価指標

Fuzzテスト試行コストの測り方



- テスト・コストを測る
 - 「異常がすぐわかった」を 1として、
 - 他の異常が読み取れるまでのコストを、比で表す。

Fuzzテストを試行



- OSS活用WGで、fuzzテストを試行する
- 対象OSS
 - OpenCV :機械学習,画像処理
 - ニーズ高い
 - OpenRTM-aist
 - 国産OSS
 - ロボット用OSS

OpenCVにFuzzテスト試行した(2016年)



- 機械学習 (認識)に関する関数をテスト
- `void cvReleaseImage( IplImage** image )`
- `void cvReleaseHaarClassifierCascade( CvHaarClassifierCascade** cascade )`
- `int cvNamedWindow( const char* name, int flags=CV_WINDOW_AUTOSIZE )`
- `CvCapture* cvCreateFileCapture( const char* filename )`
- `CvCapture* cvCreateCameraCapture( int index )`
- `int cvSetCaptureProperty( CvCapture* capture, int property_id, double value )`
- `bool CascadeClassifier::load(const string& filename)`
- `CvMemStorage* cvCreateMemStorage( int block_size=0 )`
- `IplImage* cvQueryFrame( CvCapture* capture )`
- `void detectMultiScale( const Mat& image, vector<Rect>& objects, double scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size())`
- `void cvRectangle( CvArr* img, CvPoint pt1, CvPoint pt2, CvScalar color, int thickness=1, int line_type=8, int shift=0 )`
- `void cvShowImage( const char* name, const CvArr* image )`
- `void cvReleaseMemStorage( CvMemStorage** storage )`
- `void cvReleaseCapture( CvCapture** capture )`
- `void cvDestroyWindow( const char* name )`

OpenCVにFuzzテスト試行した結果(2016)



大きな問題がありそうなもののみを、表に掲げる

関数	引数	値	結果(大きな問題がありそうなもののみ抜粋)
CvCapture* cvCreateFileCapture(const char* filename)	filename	NULL	Segmentation fault
IplImage* cvQueryFrame(CvCapture* capture)	capture	NULL	静止画使用時は黒い表示、カメラ使用時はNULLを返す
void detectMultiScale(const Mat& image, vector<Rect>& objects, double scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size())	scaleFactor or	inf	呼び出してから戻ってこない
void cvShowImage(const char* name, const CvArr* image)	name	存在しないウィンドウ名	(Hankyu+Face Detect:12853): GLib-GObject-CRITICAL **: g_object_unref: assertion 'G_IS_OBJECT (object)' failedを表示し、新たなウィンドウ生成
	image	NULL	真っ黒の表示。(Hankyu+Face Detect:16287): GLib-GObject-CRITICAL **: g_object_unref: assertion 'G_IS_OBJECT (object)' failed表示。

OpenCVにFuzzテスト試行した(2016)

- ファースト・インプレッション
 - OpenCVって、なかなか品質が高い(主観的感想)
 - でも、バグを見つけたぜ!
 - 異常値の対応不備。通常使用では、問題にならない

OpenRTM-aistにFuzzテスト試行した(2017年)



- 関数/対象 引数/設定項目
 - コンフィギュレーション・パラメータ
 - bindParameter関数 param_name
 - RTC::RingBufferのコンストラクタ length(long int,バッファ長)
 - RTC::Properties::load 入力ストリームへ与える文字列(キー:値)
 - RTC::Manager::createComponentcomp_args(const char *)
 - RTC::Manager::initargc, argv
 - RTC::RTObject_impl::setInstanceName instance_name(const char *)
-
- rtc.conf中のネームサーバへの登録名

※もともと、RTMは、通信の枠組みなので、設定可能なパラメータが少ない

OpenRTM-aistにFuzzテスト試行した結果(2017)



問題がありそうなもののみを、表に掲げる

関数/対象	引数/設定項目	値	得られた結果 (怪しいもののみ)	備考
コンフィギュレーション・パラメータ(coil::Propertiesのコンストラクタ、char* []引数でキー・値の並びを渡す)	キー名	キー名: nullptr	Segmentation Fault	
bindParam関数	param_name	nullptr	0	第3引数で指定したデフォルト値(10)とならない。
RTC::RingBufferのコンストラクタ	length(long int, バッファ長)	-1	Component create failed. 中止 (コアダンプ)	仕様では、2以下を指定するとバッファ長2で初期化される、となっている。
		0	length()が0	仕様では、2以下を指定するとバッファ長2で初期化される、となっている。
		1	length()が1	仕様では、2以下を指定するとバッファ長2で初期化される、となっている。
RTC:Properties::load	入力ストリームへ与える文字列(キー: 値)	キー・値共にaを3173回繰り返し	値だけが923文字に切り詰められる。また、例外送出されることがあった。(17回中2度発生した)。	terminate called after throwing an instance of 'CORBA::BAD_INV_ORDER'
RTC::RTObject_impl::setInstanceName	instance_name(const char *)	nullptr	Segmentation fault	



OpenRTM-aistにFuzzテスト試行した(2017)



- ファースト・インプレッション
 - 品質が悪くない(主観的感想)
 - 問題が出たのも、初期化で使用するところ
- でも、バグを見つけたぜ!
 - 異常値の対応不備。通常使用では、問題にならない



- ソフトウェアごとのバグの出方の傾向を分析してみたい
 - 今年度中に
- これまでの試行対象(2件)は、品質が高すぎた印象
 - *雑な* OSSを対象にやってみたい
 - 品質の低いOSSの傾向をみたい
- コスト、指標づくりなどの分析は、これから
 - 2016,2017に「雑なOSS」の結果を加えて分析する
- OSS活用WGへのご参加を!
 - 一緒に、OSS品質評価問題に取り組みましょう!



「OSSの品質についてのアプローチ 2018」

2018/11/14 発行

発行者 一般社団法人 組込みシステム技術協会
東京都中央区日本橋大伝馬町 6-7
TEL: 03(5643)0211 FAX: 03(5643)0212
URL: <http://www.jasa.or.jp/>

本書の著作権は一般社団法人組込みシステム技術協会（以下、JASA）が有します。

JASAの許可無く、本書の複製、再配布、譲渡、展示はできません。

また本書の改変、翻案、翻訳の権利はJASAが占有します。

その他、JASAが定めた著作権規程に準じます。