ひろじれん　プロジェクト

Version 0.08

01/07/2019

太田　薫

**目次**

1 はじめに 5

2 ひろじれんプロジェクト 6

2.1 プロジェクトとは 6

2.1.1 プロジェクトの目的と成功の条件 6

2.1.2 開発プロセス 6

2.2 ひろじれんプロジェクトの目的と成功の条件 7

2.2.1 成果物 7

2.2.2 プロジェクト・メンバー 7

2.2.3 トレードオフマトリックス 8

2.3 やってみたい事リスト 8

2.4 目標を実現するための課題 9

2.4.1 目標「大学で学んだ制御則を実機に搭載する」の課題 9

2.4.2 目標「地区大会でリザルトタイム0秒以下を実現する」の課題 9

2.4.3 目標「モデルシートでA以上の評価を得る」の課題 9

2.4.4 目標「リーン開発の本質を身につける」の課題 9

2.4.5 目標「「画像処理で制御を行う移動型ロボット」を復習を行い、ロボット工学研究への復帰を目指す」の課題 9

2.4.6 目標「“開発”のやり方を学び、最近悩んでいる「行き当たりばったり」の業務を改善する目標を実現するための課題」の課題 9

2.5 プロジェクト・マネジメント 10

2.5.1 プロジェクト・マネジメントの基礎概念 10

2.5.2 プロジェクト・マネジメントのプロセス 12

2.5.3 ものごとの重要度について 13

3 リーン開発 14

3.1 リーン開発とは 14

3.2 リーンソフトウェア開発の七つの原則 14

3.2.1 原則1　ムダを無くす(Eliminate Waste) 14

3.2.2 原則2 品質を作り込む(Build Quality In) 15

3.2.3 原則3:知識を作り出す(Create Knowledge) 15

3.2.4 原則4:決定を遅らせる(Defer Commitment) 15

3.2.5 原則5:速く提供する(Deliver Fast) 16

3.2.6 原則6:人を尊重する(Respect People) 16

3.2.7 原則7:全体を最適化する(Optimize the Whole) 16

3.3 ○ 16

4 ひろじれんシステム 17

4.1 システムとは、 17

4.1.1 システムのライフサイクル 17

4.1.2 システムのコンセプト 17

4.1.3 Systems Engineeringのテクニカル・プロセス 18

4.1.4 開発プロセスと手法 19

4.2 システムの概要 20

4.2.1 ひろじれんシステムの目的と目標 20

4.2.2 システムのライフサイクル 20

4.2.3 ステークホルダー 21

4.2.4 システム構成 22

4.2.5 コンセプト 22

5 システム設計 23

5.1 システム設計とは 23

5.2 要求分析 24

5.2.1 要求とは 25

5.2.2 要求分析に用いられる図やモデル　~~要求分析の成果物~~ 27

5.2.3 コンテクスト分析 28

5.2.4 ユースケース分析 29

5.3 あなごシステムの要求分析 30

5.3.1 ステークホルダー要求 30

5.3.2 システムの振る舞い(シナリオ) 31

5.3.3 システムの要求 32

6 Appendix 33

6.1 Matlab Simulinkモデルの実装方法 33

**変更履歴**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Description** | **Who** |
| 0.01 | 04/06/2019 | * ドラフト | 太田 |
| 0.02 | 04/07/2019 | * 目標とやってみた事リストの追加 | 太田 |
| 0.03 | 04/16/2019 | * リーン開発の章を追加 | 太田 |
| 0.04 | 04/21/2019 | * くりすさんの目標とやってみたい事を追加 | 太田 |
| 0.05 | 04/26/2019 | * 諸田さんの目標とシステムの目標を追加 | 諸田  太田 |
| 0.06 | 05/10/2019 | * リーンソフトウェア開発の七つの原則を追加 | 太田 |
| 0.07 | 05/25/2019 | * 目標を実現するための課題を追加 | 太田 |
| 0.08 | 01/07/2019 | * Matlab Simulinkモデルの実装方法を追加 | 太田 |

# はじめに

hogehoge。

# ひろじれんプロジェクト

## プロジェクトとは

「プロジェクトとは、独自のプロダクト、サービス、所産を創造するために実施される有期性の業務である」

PMBOK

|  |  |  |
| --- | --- | --- |
|  | 特徴 | 説明 |
| 1 | 有期性 | 明確な開始日と終了日がある |
| 2 | 独自の成果物 | 以前に作成されたことが無い、独特な製品やサービスを開発できる |
| 3 | 繰り返し性がない | 同じプロジェクトを二度行う事はない |
| 4 | 段階的な詳細化 | 作成される製品やサービスは徐々に段階を追って詳細化される |
| 5 | 複数のメンバー | プロジェクトマネージャと複数のメンバーによって実行する |

**プロジェクトの三大要素**

|  |  |
| --- | --- |
| 要素 | 説明 |
| スケジュール | プロジェクトの時間的要素(いつ完了するか) |
| リソース(コスト) | プロジェクトの金額的な要素(いくらかかるか) |
| スコープ | プロジェクトの実施範囲(どこまでやるか、どのレベルまでやるか) |

### プロジェクトの目的と成功の条件

プロジェクトの成功とは何をもって成功というのか？

プロジェクトから獲得すべき価値を明確にした上でプロジェクトの目的を明らかにし、成功の定義を定める必要があります。

### 開発プロセス

|  |  |
| --- | --- |
| 開発プロセス | 特徴 |
| ウォーターフォール型開発(V字型モデル) | 比較的大規模なプロジェクト向け、工程ごとの成果物が明確、後戻りが想定されていない |
| 反復型開発 | システムをサブシステム、コンポーネントに分割し、順次開発する |
| アジャイル型開発 | イテレーションを採用、開発対象を小さな機能に分割し、１つのイテレーションで１つの機能を開発 |

## ひろじれんプロジェクトの目的と成功の条件

プロジェクトの目的：**「自己の成長」**

プロジェクトの目標

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 目標 | 期日 | 成果物 | 主担当 |  |
| 大学で学んだ制御則を実機に搭載する | 2019/9/15 | 走行体システム  制御の仕様書 | 諸田・いつか | ○ |
| ｍｍ地区大会でリザルトタイム0秒以下を実現する | 2019/9/15 | 走行体システム | 太田 | ☓ |
| モデルシートでA以上の評価を得る | 2019/8/25 | モデルシート | 太田 | C- |
| リーン開発の本質を身につける | 2019/9/15 | CI環境 | 太田 | 勉強はしたが、▲ |
| 「画像処理で制御を行う移動型ロボット」を復習を行い、ロボット工学研究への復帰を目指す | 2019/11/15 |  | くりす | ○ |
| “開発”のやり方を学び、最近悩んでいる「行き当たりばったり」の業務を改善する | 2019/11/15 |  | くりす | これから  期限がある仕事のやり方 |

### 成果物

成果物

|  |  |
| --- | --- |
| 成果物 | 備考 |
| 走行体システム |  |
| モデルシート |  |
| CI環境 |  |

### プロジェクト・メンバー

|  |  |
| --- | --- |
|  | 氏名 |
|  | もろた |
|  | いつか |
|  | くりす |
| PM | おおた |

### トレードオフマトリックス

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 柔軟性  低 | 柔軟性  中 | 柔軟性  高 | 備考 |
| スコープ  (機能) |  |  | ● | プロジェクトの目的は競技の結果ではなく人材育成である。 |
| スケジュール |  | ● |  | 機能を割り切る事で開発スケジュールは調整できる。  モデルシートの締切日、地区大会の開催日は決まっている。 |
| 資源 | ● |  |  | 開発リソース　人　金は限られている |

## やってみたい事リスト

|  |  |  |
| --- | --- | --- |
| No | やってみたい事 | Who |
| 1 | カルマンフィルタによる自己位置推定 | おおた |
| 2 | ニューラルネットワークによるパターン認知 | おおた |
| 3 | モデルによるロボットの制御 | おおた |
| 4 | 継続的インテグレーション | おおた |
| 5 | 大学時代の研究の復習 | くりす |
| 6 | モデルによるロボットの制御 | くりす |
| 7 | “開発業務”の取り組み方の勉強 | くりす |
| 8 | 現物、実機による制御の勉強　(PID) | いつか |
| 9 | PID制御 | もろた |
| 10 | データ駆動 | いつか |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 目標を実現するための課題

### 目標「大学で学んだ制御則を実機に搭載する」の課題

|  |  |  |
| --- | --- | --- |
| 課題 | 期日 | 備考 |
|  |  |  |
|  |  |  |
|  |  |  |

### 目標「地区大会でリザルトタイム0秒以下を実現する」の課題

|  |  |  |
| --- | --- | --- |
| 課題 | 期日 | 備考 |
|  |  |  |
|  |  |  |
|  |  |  |

### 目標「モデルシートでA以上の評価を得る」の課題

|  |  |  |
| --- | --- | --- |
| 課題 | 期日 | 備考 |
| モデルシートの要件の明確化 | 6月15日 |  |
| ポイント、エッセンスの定義 | 6月30日 |  |
| 一貫性のあるモデルシートの作成 | 8月4日 |  |

### 目標「リーン開発の本質を身につける」の課題

|  |  |  |
| --- | --- | --- |
| 課題 | 期日 | 備考 |
| リーン開発の本質を知る | 8月4日 |  |
| リーン開発を実践し経験する | 9月15日 |  |
|  |  |  |

### 目標「「画像処理で制御を行う移動型ロボット」を復習を行い、ロボット工学研究への復帰を目指す」の課題

|  |  |  |
| --- | --- | --- |
| 課題 | 期日 | 備考 |
|  |  |  |
|  |  |  |
|  |  |  |

### 目標「“開発”のやり方を学び、最近悩んでいる「行き当たりばったり」の業務を改善する目標を実現するための課題」の課題

|  |  |  |
| --- | --- | --- |
| 課題 | 期日 | 備考 |
|  |  |  |
|  |  |  |
|  |  |  |

## プロジェクト・マネジメント

### プロジェクト・マネジメントの基礎概念

プロジェクトとは「スコープ」「リソース(コスト)」「スケジュール」の3次元の世界を目的地に向かって飛ぶ飛行機。プロジェクト・マネジメントはその飛行機を巡航する「仕組」であり「やり方」。

飛行機の巡航システム

・飛ぶべき軌道と現在の軌道との“ズレ”をリアルタイムに検知する。

・検知した“ズレ”を数値化し制御機能にフィードバックする。

・フィードバックされた“ズレ”の大きさと方向に合わせて軌道修正のための対応を実施。

プロジェクトもそのプロジェクトの本来の正しい軌道に対する「スケジュール」「リソース(コスト)」「スコープ」の3次元上のズレを正確に数値化し巡航システムにフィードバックしなければならない。ズレを数値化する手段の一つにEVM(Earned Value Management)がある(*参考文献1*)。異なる次元の数値は比較できない。例えばシステム設計の進捗1%とシステム検証の進捗1%では作業量も作業の専門性も違う。比較できないということはスケジュール遅延リスクを数字で判断できない事を意味している。 EVMではこの問題点を解決してくれる。

プロジェクトとは、「スケジュール」「リソース(コスト)」「スコープ」それぞれを一辺とする面積**一定**の三角形である。どれか一辺を長くしようとすれば、他の一辺または二辺を短くしなければならない(トレード・オフ・トライアングル)。プロジェクトの大失敗の原因の多くは経営判断の誤り。

|  |
| --- |
| スコープ  スケジュール  リソース(コスト)  プロジェクト |

複雑なプロジェクトも分解すれば、単純な作業の積み重ねになる。

*The secret of getting ahead is getting started. The secret of getting started is breaking your complex overwhelming tasks into small manageable tasks, and then starting on the first one.*

*Mark Twain*

単純な問題について時間をかけて対応することはない。複雑なままで対応しようとしても、何からどう進めれば良いか、整理できない。管理できるレベルに分解し現実的な管理を行う。

Work Breakdown Structure(WBS)

仕事(作業)を“構造的に”“漏れ無くダブり無く”(MECE)なく分割し、整理する。

WBSを作ることはWBSのドキュメントを作ることではなく、WBSを全メンバーがコミットすること、つまり“やれる”と思えること。　大事なことは形式でもツールでもなくやれると思えること。

### プロジェクト・マネジメントのプロセス

PMBOKの知識エリアと各プロセスの成果物

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| プロセス群  知識エリア | 立ち上げ | 計画 | 実行 | 監視  コントロール | 終決 |
| 4.統合  マネジメント | 4.1  プロジェクト  憲章作成 | 4.2プロジェクト  マネジメント計画書作成 | 4.3プロジェクト実行の指揮マネジメント | 4.4プロジェクト作業の監視・コントロール  4.5統合変更管理 | 4.6  プロジェクトやフェーズの集結 |
| 5.スコープ・  マネジメント |  | 5.1要求事項収集  5.2スコープ定義  5.3WBS作成 |  | 5.4スコープ検証  5.5スコープコントロール |  |
| 6.タイム・  マネジメント |  | 6.1アクティビティ定義  6.2アクティビティ順序設定  6.3アクティビティ資源見積  6.4アクティビティ所要期間見積  6.5スケジュール作成 |  | 6.6スケジュール・コントロール |  |
| 7.コスト・  マネジメント |  | 7.1コスト見積  7.2予算設定 |  | 7.3コスト・コントロール |  |
| 8.品質マネジメント |  | 8.1品質計画 | 8.2品質保証 | 8.3品質管理 |  |
| 9.人的資源マネジメント |  | 9.1人的資源計画書作成 | 9.2プロジェクト・チーム編成  9.3プロジェクト・チーム育成  9.4プロジェクト・チームのマネジメント |  |  |
| 10.コミュニケーション・マネジメント | 10.1ステークホルダー特定 | 10.2コミュニケーション計画 | 10.3情報配布  10.4ステークホルダーの期待のマネジメント | 10.5実績報告 |  |
| 11.リスク・マネジメント |  | 11.1リスク・マネジメント計画  11.2リスク特定  11.3定性的リスク分析  11.4定量的リスク分析  11.5リスク対応計画 |  | 11.6リスクの監視・  コントロール |  |
| 12.調達マネジメント |  | 12.1調達計画 | 12.2調達実行 | 12.3調達管理 | 12.4調達終結 |

### ものごとの重要度について

すべての活動は、緊急度と重要度という2つの軸によって4つの領域に分けることが出来る。大きな目標のために役立つ行動や、意義のある重要なことが私たちに働きかけることことはない。それらは「緊急」の事柄ではないからだ。自ら働きかけなければならないものである。

|  |  |  |
| --- | --- | --- |
|  | 緊急 | 緊急ではない    一番大事 |
| 重要 | 第一領域  ・差し迫った問題  ・締め切りのある仕事、会議  ・品質不具合の対応 | 第二領域  ・信頼関係の構築  ・戦略の立案、準備や計画  ・予防  ・価値観の明確化  ・勉強や自己啓発  ・人材育成 |
| 重要ではない | 第三領域  ・重要でない電話  ・重要でない報告書  ・重要でない会議  ・重要でない差し迫った問題  ・みんながやっていること  ・突然の来訪  ・無意味な付き合い | 第四領域  ・暇つぶし  ・意味のない活動  ・見せかけの仕事  ・現実逃避  ・待ち時間 |

第二領域が最も重要度の高い活動であるが、周りからは求められることは無い。自ら働きかけて実行していく必要がある。

# リーン開発

## リーン開発とは

リーン開発

* 頻繁な製品変更(ソフトウェアリリース)
* 短い開発期間
* 開発ステップ間の情報在庫の削除
* 開発ステップ間で仮決定の情報を小ロットずつ頻繁に移動させる
* 開発時間削減にはリソースのゆとりステージ間での情報フローが必要になる
* 製品設計、スケジュール、原価目標の変更への順応性
* エンジニア(開発者)への幅広いタスク割り当てによって、高い生産性が得られる
* 頻繁でインクリメンタルな革新と継続的な製品・プロセス改善の重視
* 品質、開発時間、開発生産性の同時改善

## リーンソフトウェア開発の七つの原則

原則1：ムダを無くす

原則2：品質を作り込む

原則3：知識を作り出す

原則4：決定を遅らせる

原則5：早く提供する

原則6：人を尊重する

原則7：全体を最適化する

### 原則1　ムダを無くす(Eliminate Waste)

*リーンソフトウェア開発では、価値を付加しないすべてのムダの排除によって、工数を短縮することに重点を置いている。*

*ムダとは、価値を付加しないあらゆるもののことだ。ムダをなくす第一歩として、「価値とは本来何なのか」を感じ取る感性を磨かなくてはならない。*

*80%の価値を提供する20%のコードをまず開発し、その後初めて、次に重要な機能の開発にとりかかるようなプロセス。(Implementing Lean Software Development, Mary Poppendieck & Tom Poppendieck)*

### 原則2 品質を作り込む(Build Quality In)

*目標は、最初からコードに品質を作り込むことであり、後でテストをすることではない。トラッキングシステムに欠陥情報を入力することに注力していないで、最初から欠陥を作らないようにするので。そのためには、組織にしっかしとした規律が求められる。*

*本当に品質を手に入れたいのなら、事後の検査をするのではなく、最初から欠陥が入り込まないような状況にしなくてはならない。それができないのであれば、少し進めるごとに製品を検査し、発生したらすぐに欠陥をつかまえるようにしなくてはならない。(Implementing Lean Software Development, Mary Poppendieck & Tom Poppendieck)*

Keyword:テスト駆動型開発

### 原則3:知識を作り出す(Create Knowledge)

ソフトウェア開発を成功に導く4つのプラクティス

1. 最小の機能セットを早期に顧客にリリースし、評価とフィードバックを得ること
2. 毎日ビルドし、統合テストからすばやいフィードバックを得ること
3. 経験と直感を備えたチームやリーダーに適切な決定を下させること
4. 新規機能が追加しやすいモジュールアーキテクチャにすること

*製品開発において、長期的に卓越した能力を見せる企業には、共通点がある。それは、規律に従った実験を通じて新たな知識を生み出し、その知識をより大きな組織内で参照できるようにきりんと文章としてまとめる。というて点である。そのような企業は、明確なデータを記録するだけではなく、暗黙知を明確化する方法を見つけて、それもまた組織的なナレッジベースの一部にしている。また、開発中の製品について学ぶことも重要だが、その一方で、その知識を将来の製品に焼くに立てるために文章にしておくことが絶対必要であることも理解しているのである。*

*開発サイクルを通じて、体系的な学習を促す開発プロセスを用いることは重要だが、その開発プロセス自体も、体系的に改善していかなくてはならない。どのようなチームであっても、プロセス改善のための時間を、通常の業務とは分けて、定期的に設けるべきである。(Implementing Lean Software Development, Mary Poppendieck & Tom Poppendieck)*

### 原則4:決定を遅らせる(Defer Commitment)

*難しい決断であれば、それを片付け、目の前のリスクに対処し、未知の要素を減らしたいと多くの人が思う。しかしながら、不確実性と対面する場合、それが複雑性を伴うものであればなおさら、決定を下さなくてはならない瞬間まで重要な選択肢をオープンのままにしておりて、いろいろなソリューションを実験して難しい問題に対処するのがいちばんである。実際、最高のソフトウェア設計戦略の多くは、取り返しのつかない決定をできるかぎり遅く下せるように、選択肢をオープンにしておくことに照準を合わせている。(Implementing Lean Software Development, Mary Poppendieck & Tom Poppendieck)*

### 原則5:速く提供する(Deliver Fast)

*品質を作り込まなければ、ハイスピードは維持できない。*

*高品質を達成するにま、2つ方法がある。ひとつは、スピードを落として慎重になる。という方法である。もうひとつは、プロセスを継続的に改善し、製品に品質を作り込む人を育て、競合他社の何倍ものスピードで繰り返し、確実に顧客の要求に反応する能力を培う、という方法だ(Implementing Lean Software Development, Mary Poppendieck & Tom Poppendieck)*

### 原則6:人を尊重する(Respect People)

1. 起業家的リーダー
2. エキスパートエンジニアの尊重
3. 責任ベースのプランニングと制御

### 原則7:全体を最適化する(Optimize the Whole)

*ソフトウェア開発は、伝統的に部分最適に陥りやすい*

*組織の境界をまたげば、コストがかかる。ピーター・ドラッガーは、バリューストリームの始めから終わりまでの全体を、ひとつのマネジメントシステムで対応している企業では、競合他社に比べ、25〜30%のコスト削減ができていると述べている。(Implementing Lean Software Development, Mary Poppendieck & Tom Poppendieck)*

## ○

# ひろじれんシステム

## システムとは、

システムとは、定義された目的を実現するために必要な要素の集まり。要素はソフトウェア、ハードウェア、人、サービスなどである。

*Systems are man-made, created and utilized to provide products or services in defined environments for the benefit of users and other stakeholders.*

ISO/IEC/IEEE 15288

*System is an integrated set of elements, subsystems, or assemblies that accomplish a defined objective.These elsements include products(hardware,software,firmware),processes,people,information,techniques,facilities,services,and other support elements.*

INCOSE

### システムのライフサイクル

システムのライフサイクルを定義し、

・開発の流れを把握する

・競技だけではなく、ライフサイクル毎のシステムへの要求を洗い出し、仕様の抜け漏れを防ぐ

・開発に関わる全ての利害関係者を洗い出す

Generic life cycle (ISO/IEC/IEEE 15288:2015)

Concept

Stage

Development Stage

Production

Stage

Utilization Stage

Support Stage

Retirement

Stage

### システムのコンセプト

コンセプトを立案し、成果物のイメージを明確にする。次にシステムの目的を達成するために実現したい振る舞いを明らかにする。

### Systems Engineeringのテクニカル・プロセス

*The Technical Processes are used to define the requirements for system, to transform the requirements into a effective product, to permit consistent reproduction of the product where necessasy, to use the product to provide the required services, to sustain the provision of those services and to dispose of the product when it is retired from service.*

ISO/IEC/IEEE 15288

*Technical processes enable systems engineers to coordinate the intractions between engineering specialists, other engineering disciplines, system stakeholders and operators, and manufacturing. They also address confrmance with the expectations and legislated requirements of society. These processes lead to the creation of a sufficient set of requirements and resulting system solutions that address the desired capabilities within the bounds of performance, environment,external interfaces,and design constraints.*

INCOSE

|  |
| --- |
|  |

### 開発プロセスと手法

ETロボコンで経験できる箇所

開発・設計に必要な人、プロセス、ツール、手法、技術、情報を下記に示す。



## システムの概要

数値化されていますか？

### ひろじれんシステムの目的と目標

根拠は明確ですか？

目的：「チャレンジの場の提供」

目標①：リザルトタイム0秒を獲得する。

走行タイム :20秒

カラーブロック有効移動8個 :-4秒

フルビンゴ :-11秒

何を、どれだけ達成しますか？

ガレージ停止 :-5秒

目標②：成功率100%

正確に走行体を制御する

目標③：実験・検証環境を提供する

ハードルの高い目標となっている。

5,7月末に目標の見直しを行う。

2019/4/26

### システムのライフサイクル

|  |
| --- |
| 引き継ぎ  ソフト設計(モデル作成)  ソフト実装・検証  システム検証  地区大会  技術フォーラム  システム設計  ロボット特性  の把握  コンセプト立案 |

### ステークホルダー

|  |  |
| --- | --- |
| ステークホルダー | 何が欲しいのか |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

### システム構成

### コンセプト

|  |  |
| --- | --- |
| システムの目的 | チャレンジの場の提供 |
| 目標① | リザルトタイム0秒以下を獲得 |
| 目標② |  |
| 目標③ |  |

# システム設計

## システム設計とは

ステークホルダの要求を要求分析しシステム要件を定義、要件を満たすアーキテクチャ設計を行い、サブシステムへの要求（システム仕様書）を導く活動。

下記にシステム設計のプロセスと手法を示す。

|  |
| --- |
|  |

## 要求分析

要求分析とは、ステークホルダの要求をシステムの要件へと変換する作業である。

システムの境界と成果物を明確にする作業である。

下記に要求分析のプロセスを示す。

|  |
| --- |
|  |

### 要求とは

*製品が「何を、どれほどよく、どういった状態で」、与えられた目的を達成するかを決めるもの合意、計画、仕様のような標準となる文章*

ANSI/EIA-632

|  |
| --- |
| 要求  機能  仕様  仕様  機能の品質  仕様  要求  機能  仕様  仕様  機能の品質  仕様  目標 |

**要求の種類**

|  |  |
| --- | --- |
| 要求 | 説明 |
| 機能 | システムが目的を実現するために持つべき機能に対する要求 |
| 性能 | システムが目的を実現するために持つべき性能に対する要求 |
| 外部インタフェース | システムと外界とのインタフェースに対する要求 |
| 環境 | システムが動作する環境に関する要求 |
| リソース | システムが目的を実現するためにリソースに関する要求 |
| 物理的 | システムに対する物理的要求(容積、サイズ、重量) |
| 設計 | 設計基準の要求 |
| プロセスと手法 | 製造プロセスおよび品質管理要求 |

**ISO9126:品質特性　機能要求と非機能要求　(ソフトウェアの品質要求)**

|  |  |
| --- | --- |
| 機能要求 | 非機能要求 |
| 機能性要求 | 信頼性要求  使用性要求  効率性要求  保守性要求  移植性要求 |

要求には機能要求と非機能要求がある。

機能要求はシステムが「何を」を実行する必要があるかを示す。

非機能要求は機能上の局面が「どの程度」達成されるかを示す。

**要求の品質**

|  |  |
| --- | --- |
| 要求の品質 | 説明 |
| 完全性 | 各要求は、当該要求を実現するために必要となる、制約および条件を含んだ、全ての情報を含んでいること |
| 無矛盾性 | 要求は、矛盾が無いこと |
| トレース可能性 | 要求は、上位および下位要求がトレース可能であること |
| テスト可能性 | 要求は、検証可能であること |
| 単一性 | 要求は、二重定義されていないこと |

**品質要素**

|  |  |
| --- | --- |
| 品質要素 | 説明 |
| 当たり前品質 | できていて当たり前、できなければ不満を引き起こすもの |
| 一元的品質 | できていれば満足、できていなければ不満を引き起こすもの |
| 魅力的品質 | できていれば満足、できていなくても仕方がないと受けとられるもの |

製品品質特性(ISO/IEC25000シリーズ)

|  |  |
| --- | --- |
| 製品品質特性 | 説明 |
| 機能適合性 | 機能が利用者のニーズにどれくらい合っているか |
| 性能効率性 | システムの実行時間が効率的か、資源の使い方が効率的か |
| 互換性 | ほかのシステムと適切につながるか、共存できるか |
| 使用性 | システムがどの程度使いやすいか |
| 信頼性 | 不具合が起きない度合い |
| セキュリティ | 情報資産へのアクセス制御、保全性を保つ度合い |
| 保守性 | システムの保守、修正が容易か |
| 移植性 | 違う環境へシステムを移すことがどの程度容易にできるか |

### 要求分析に用いられる図やモデル　~~要求分析の成果物~~

~~要件定義書~~

~~・システムが持つ機能とその機能の仕様が明確になっている事。~~

~~・制約や性能・品質の要件が明確になっている事。~~

用いられる図やモデル、表、記述法

|  |  |  |
| --- | --- | --- |
| 項目 | Systems Engineering (SysML) | UML |
| 要求図 | ● |  |
| D-CASE |  |  |
| マインドマップ |  |  |
| ユースケース図 | ● | ● |
| ユースケース記述 |  |  |
| コンテキスト図 | ● |  |
| シナリオ/基本フロー |  |  |
| アクティビティ図 | ● | ● |
| 図解 |  |  |
| 機能一覧表 |  |  |
| ミスユースケース図 | ● | ● |
| ミスユースケース記述 |  |  |

### コンテクスト分析

ライフサイクルのステップ毎にコンテクスト図を作成することで、外部から受ける影響、外部に与える影響を把握する。

・カテゴリごとにアイテムを抽出。　力学的、電気的、電磁気的、作業的、対環境など

・インターフェース条件を書き込む。

|  |
| --- |
| 競技ステージ |

### ユースケース分析

システムが利用者に提供すべき働きやサービスをモデル化します。

|  |
| --- |
|  |

ユースケース記述の例

|  |  |
| --- | --- |
| UC名 | 光環境のキャリブレーションをする |
| 概要 | 走行に必要な各種センサ値を取得し保存する |
| アクター | ユーザー |
| 事前条件 | ユーザーが走行するコースを選択していること |
| 事後条件 | 白黒基準値に異常がなく保存が完了していること |
| 基本系列 | 1.ユーザは、白線上に走行体を設置し、タッチセンサを押下する  2.システムは、光センサ値を取得し白色基準値として保存する  3.ユーザは、黒線上に走行体を設置し、光センサを押下する  4.システムは、光センサ値を取得し黒色基準値として保存する  5.システムは、白色基準値が黒色基準値より大きいことを 確認する |
| 代替系列 | プリセットされた白黒基準値を使用する |
| 例外系列 | 基本系列5で白色基準値が黒色基準値以下であった場合、 LEDやスピーカーを用いてユーザに警告を行い、基本系列1に 戻る |

## あなごシステムの要求分析

### ステークホルダー要求

### システムの振る舞い(シナリオ)

### システムの要求

# 振り返り

## 新しく経験したこと

Gitによる分散バージョン管理 Github Desktop

GitHub

Clone, commit, push, pull

Branch

Pull request

Conflict

リポジトリ

コラボレイター

構成管理

JIRAによるプロジェクト管理

スプリント　一週間単位のタスクの集まり　リーン開発　アジャイル

Cygwin UNIX系のOS

emacs

Toppers リアルタイムOS 周期タスクの管理

タスクの実行される時間とモデルの実行する時間の関係を知った。

UML, SysML

走行体モデルを作成し実装した

Matlabでモーターモデルと制御モデルを作成した。

極配置法を使って制御の係数を求めた。　初期のパラメーターを決めた。

システムを安定側に保った。

オートコードして実装した。

状態遷移図を設計して実装した。

Pythonによるコーディングが効率的

CNN

A\*探索

## 出来なかった事

・デザインレビューが出来なかった。　最低月1回やりたかった。

・リーン開発　活かしきれなかった。

## KPT

## Keep

## 良かった点　これも続けたい事

・一つの問題に対し複数の案をだせた。

機能を実現するために、失敗を繰り返しながら、複数の制御則を試した。その中から、、、

・制御則毎の特徴を見ることができた。

・リーン開発が学べた。

・実機で学べた。　シミュレーションだけでは学べないことが多かった。

・工数管理がちゃんと出来て、プロジェクトが管理できて。割り切るところは割りっきた。

・いろいろ新しい事を試せた。

## Problem問題点

・分からない事が聞きづらかった。　気楽に確認できる雰囲気じゃなかった。

・自主性が発揮しづらかった。（カンバンボード以外のタスクに目がいかない、

・タスクの全体像、先々のタスクを示せなかった。

・タスクに対してスキルが不足していた。

Cygwinのインストールなど、ダウンロードが出来なかった。　モチベーションが下がる。

・自分の出来る範囲のタスクのみやっていた。

・教育の時間がなかった、C++、モデル、ニューラルネットワークの勉強

・デザイン・レビューが出来なかった。

## Try 改善案

・GitHUbで論文の管理。

・モーターのシステム同定。

・タスクに着手する前に内容を確認する。

・自己の成長につながる事柄を優先的に行う。

・メンバーにあったタスクを割り当てる。

・ある程度メンバーに任す、放っておく。（自主性をださせるため）

自己位置推定のためのカルマンフィルタ

RGB-HSV　変換関数を救った。 Cの関数

けどコミットが出来なかった。

# Appendix

## Matlab Simulinkモデルの実装方法

1. コードに変換するモデルをサブシステムにする

|  |
| --- |
|  |
|  |

1. ブロックパラメーターのAtomicサブシステムとして扱うを有効にする

|  |
| --- |
|  |

1. Embedded Coderクイックスタートを起動

|  |
| --- |
|  |

1. サブシステムを選択

|  |
| --- |
|  |

1. 生成コードの出力にまるまるC++コードを選択

|  |
| --- |
|  |

1. ターゲットハードウェアにARM Compatible ARM 9を選択

|  |
| --- |
|  |

1. コードを生成

走行体(マイコン)に実装するのはmotor\_ctl.hとmotor\_ctl.cpp

|  |
| --- |
|  |

1. Makefileとコンフィグファイルにmotor\_ctl.oを追加

|  |  |
| --- | --- |
|  |  |

1. 未使用のヘッダファイルをコメントアウト

|  |
| --- |
|  |

1. オブジェクトの生成

|  |
| --- |
|  |

1. 関数の呼び出し

|  |
| --- |
|  |

1. コンパイル