

# 1. 要件分析 安全に走行するために必要な要件を抽出する

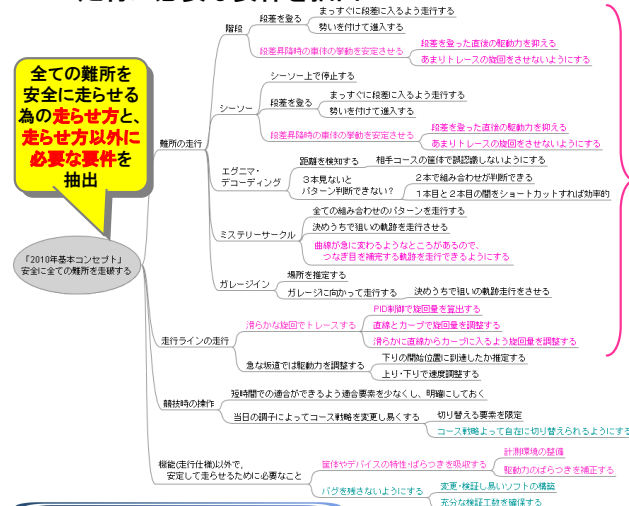
## ■ 目標(コンセプト): コース全域(難所を含む)を安全に走破する

### 〔Ⅰ〕コンセプト実現の為の要件分析

※ コンセプトからコース全域の安全な(=安定した)走行に必要な要件を抽出

### 〔Ⅱ〕非機能要件の分析

※ 〔Ⅰ〕の要件分析より、安定して走行させる為の非機能要件を『走行性能の要件』と『ソフトウェア設計品質の要件』の両面からSysMLを使って分析



安全に(=安定して)走らせるための走行性能の要件 (ピンクの文字)

抽出した制約・要件のレイヤ

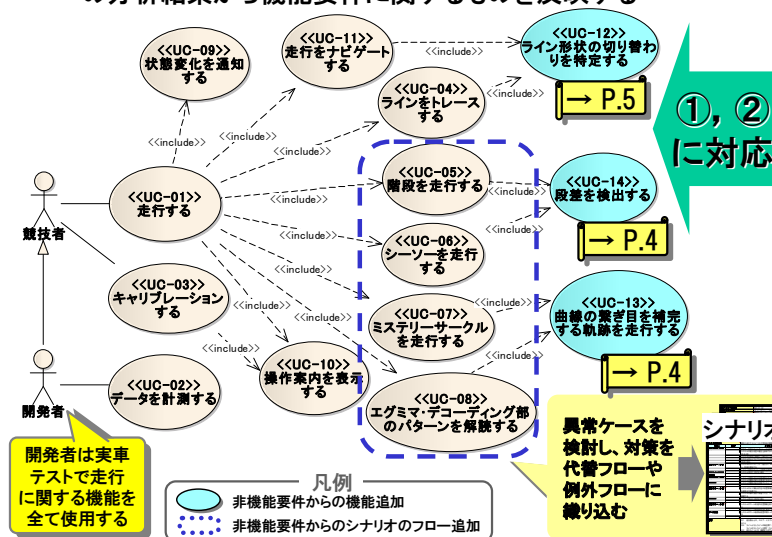
抽出した制約・要件への対策方針のレイヤ

ソフトウェアの設計品質の要件 (水色の文字)

まとめる

### 〔Ⅲ〕機能要件

※ 〔Ⅰ〕の要件分析で抽出した機能要件に、非機能要件の分析結果から機能要件に関するものを反映する



### <制約・要件への対応方針>

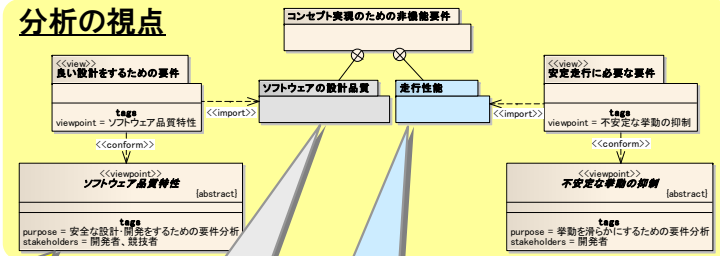
- ① 機能追加 → P.1
- ② シナリオの代替・例外フローの検討 → P.1
- ③ タスクの設計 → P.3
- ④ アーキテクチャ設計 → P.2
- ⑤ コーディング規約 → P.5
- ⑥ 命名規約 → P.5
- ⑦ 開発環境 → P.5

①, ② に対応

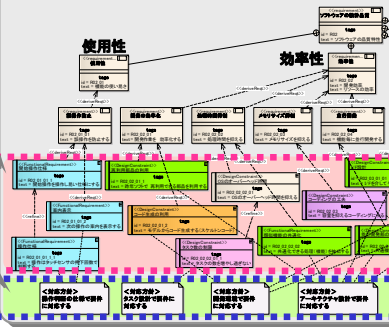


異常ケースを検討し、対策を代替フローや例外フローに絡り込む

### 分析の視点

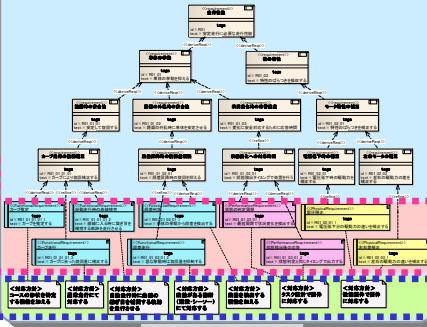


### ソフトウェア設計品質面の分析

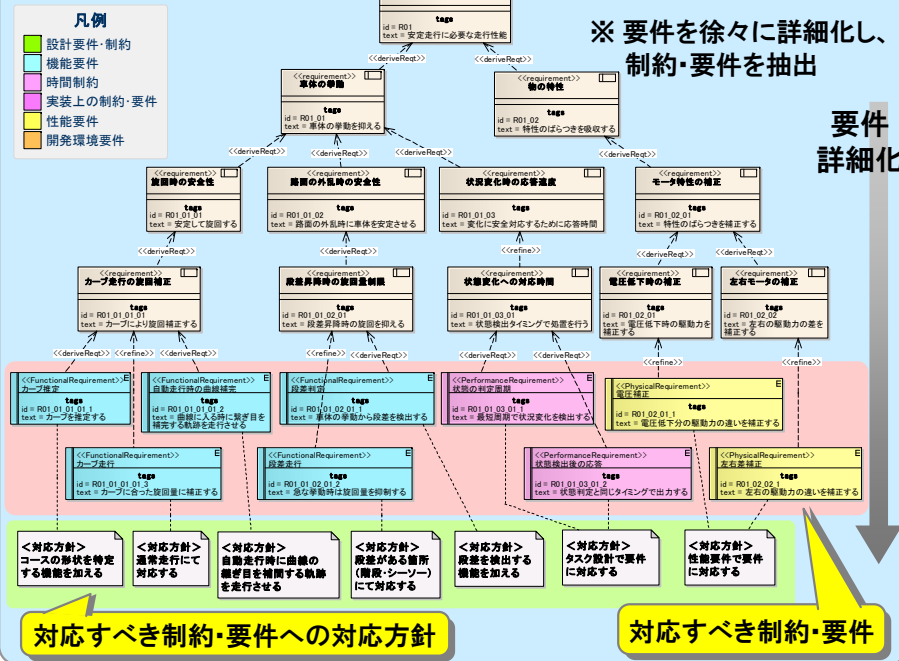


ソフトウェアの品質特性(ISO/IEC9126)から、品質のよい設計をするための要件を考える

### 走行性能面の分析



### 走行性能面の内容



※ 要件を徐々に詳細化し、制約・要件を抽出

要件詳細化

対応すべき制約・要件への対応方針

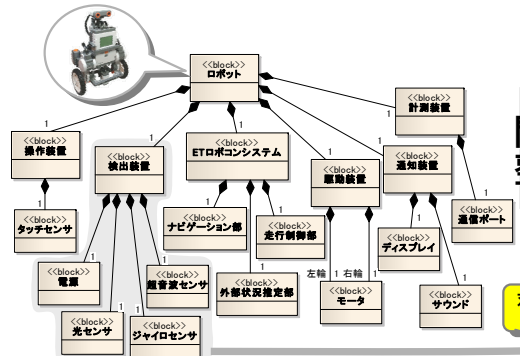
対応すべき制約・要件

## 2. 全体設計 全体のソフトウェア構造と主要なクラスの役割説明

### 〔Ⅰ〕全体構造概要

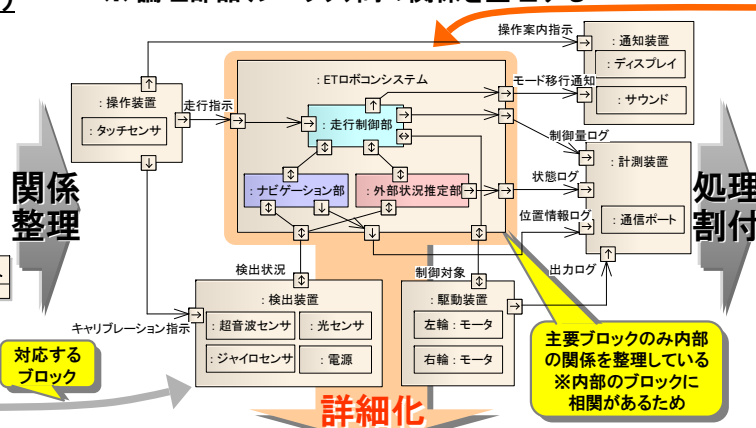
#### 全体の論理構成 (SysML:ブロック図)

※ ざっくりと必要な責務(機能)を抽出する



### 全体の論理部品間の関係 (SysML:内部ブロック図)

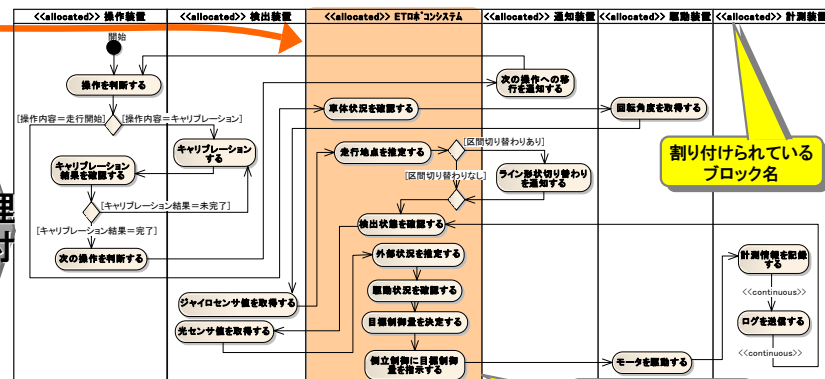
※ 論理部品(ブロック)間の関係を整理する



### 〔Ⅱ〕全体の処理手順

#### 全体の処理手順の概要

(SysML:アクティビティ図)



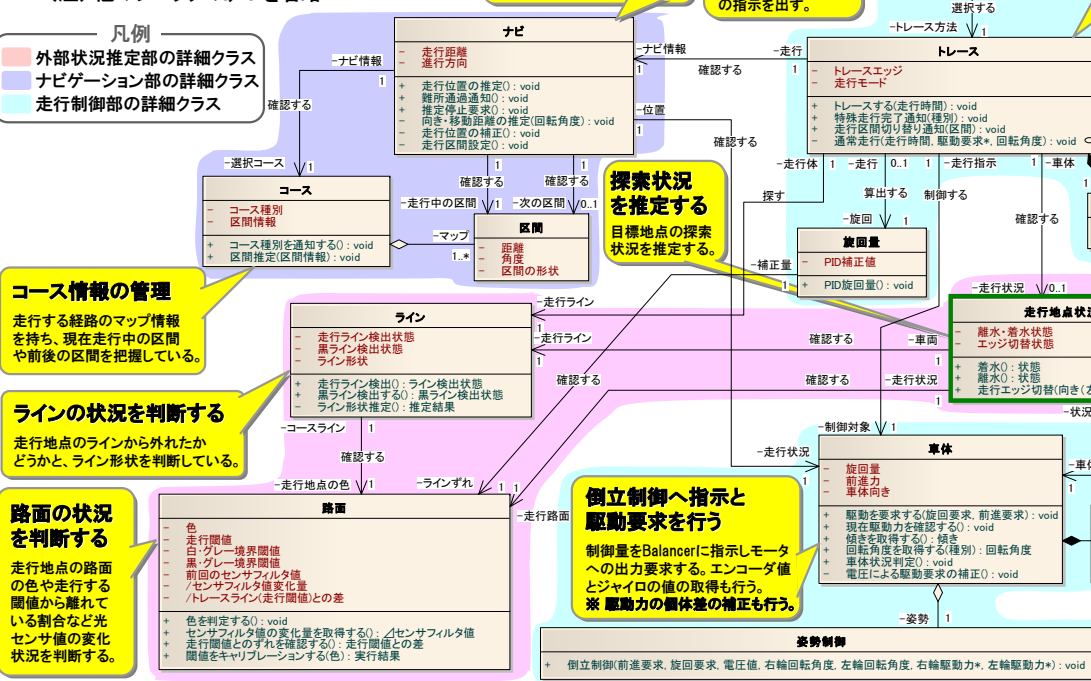
※ 全体の大まかな処理手順を考えながら  
ブロックに割り当てる処理(責務)を決める

ETロボコンシステム  
に割り付けた処理

### 〔Ⅲ〕主要部分の詳細構造

#### ※ ETロボコンシステム部の具象モデル

(注) 他のブロックのI/Fを省略



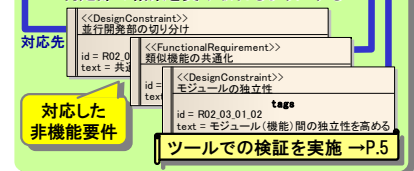
#### ライトレースの走行判断と難所の特殊走行の切替えを行う

- ・ラインから外れた度合いに応じてPID制御による制御量を判断する。
- ・ナビで現在位置を確認して難所の入口に到達したら、各難所の走行を指示する。
- ※坂道の加減速や旋回量の調整もここで行う

切替えの状態遷移図 → P.3

#### <アーキテクチャ設計方針>

1. クラス間の関連を少なくする  
対応策  
・性質によって情報をまとめる  
・関連が増える場合は、情報の受け渡し方や構造を見直す
2. 双方向の関連は原則として禁止する  
対応策  
・双方向の関連が生じた場合は、情報の受け渡し方や構造を見直す  
・例外で逆方向を許可する場合は、通知処理のみとする
3. 類似機能(処理)を共通化できるようにする  
対応策  
・開発中に増える可能性がある共通処理は場所や状況の推定ロジックと考え、状況推定するクラスを作る
4. 並行開発する難所走行部は通常走行と切り離す  
対応策  
・難所毎にクラスを分け、かつ共通するライトレースの制御量は、通常走行判定部の結果を受け取れるようにする

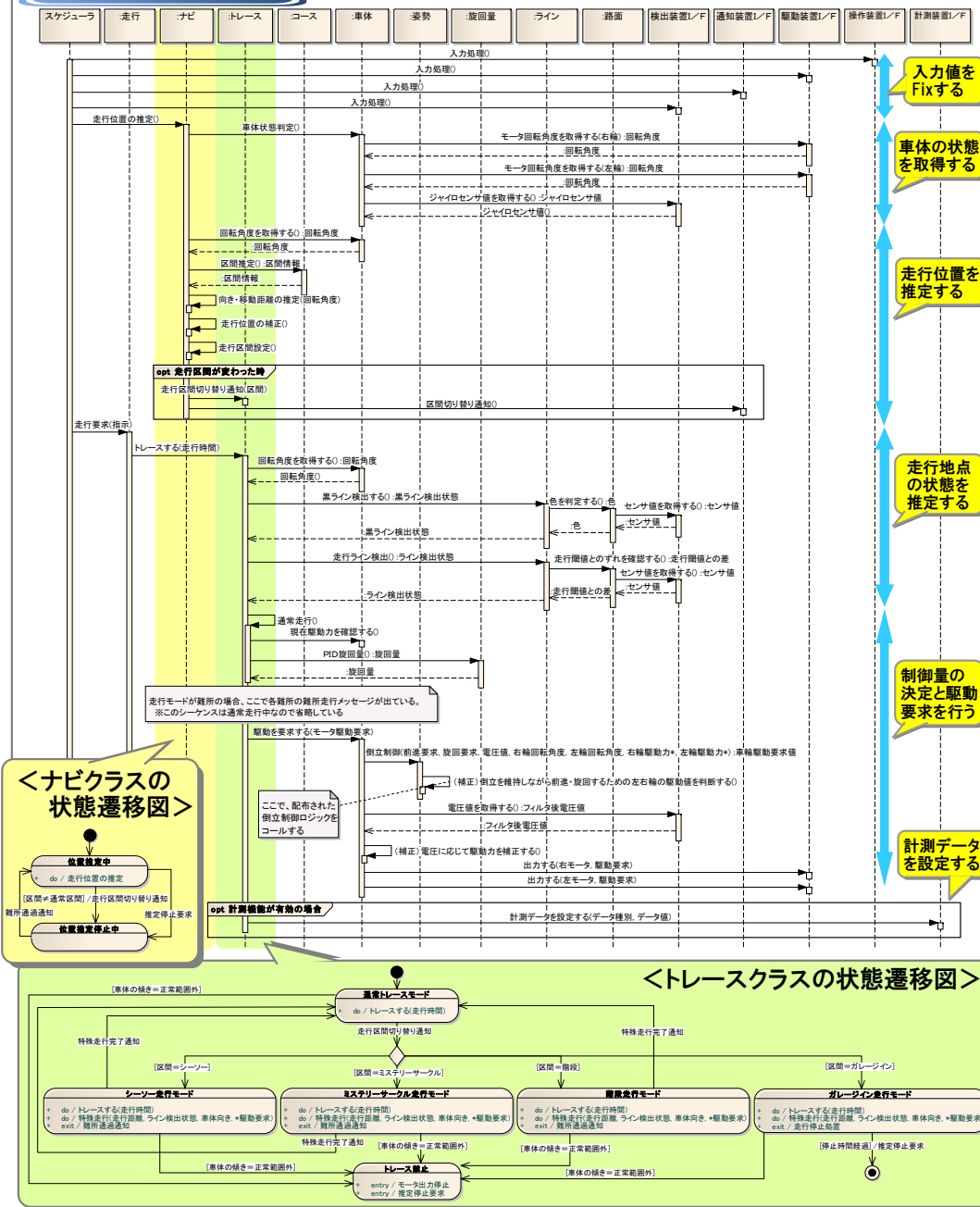


### 3. 処理の流れ

## 全体の処理の流れとタスク設計

#### 〔Ⅰ〕全体の流れ

例) 主要部分(ETロボコンシステムブロック)の通常走行時のシーケンス



#### 〔Ⅱ〕タスク設計と並行性

##### タスク分割 (SysML:ブロック図)

##### <タスク分割設計時の検討事項>

- ③ タスクの数を増やしすぎたくない (OSのオーバーヘッドが増えるため処理時間が增加する)
- ③ 操作装置は走行前の処理で走行中の処理負荷はないので制御部と同じでも良い
- ② 同期して実行させたい処理は同じタスクに割り付ける
- ② 時間がかかる又は長い演算周期で良い処理は別タスクにして、制御部の処理負荷を軽減する
- ① 本来の機能以外の負荷機能(開発時だけ使用して競技時に使用しない機能)を別タスクにして、負荷機能使用時にも制御部の処理負荷を競技時と同等にする

※ 論理部品(ブロック)のタスクへの割り付けを検討し、各タスクの時間制約を検討する

##### <対応すべき要件>

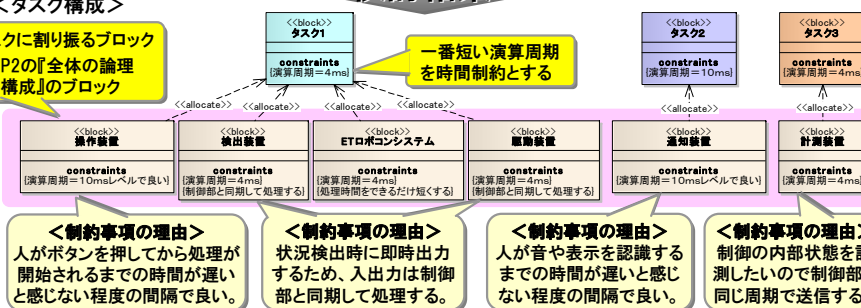
- ① <PerformanceRequirement> 状態検出後の応答  
tag: id = R01.01.03.01  
text = 状態判定と同じタイミングで出力する
  - ② <PerformanceRequirement> 状態の判定周期  
tag: id = R01.01.03.01  
text = 最長判定と同一タイミングで出力する
  - ③ <DesignConstraint> タスク数の制限  
tag: id = R02.02.03.01  
text = タスクの数を増やし過ぎない
- <対応方針>  
操作装置を制御部と同じタスクに割り付ける
- <対応方針>  
通知操作を別タスクとする
- <対応方針>  
計測操作を別タスクとする

##### 検討結果

##### <タスク構成>

タスクに割り振るブロック

※P2の「全体の論理構成」のブロック

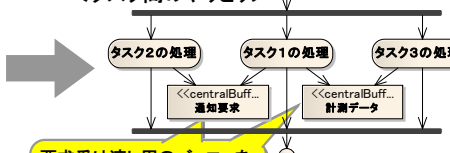


##### タスクの優先順 ※タスクの責務と時間制約から優先度を決める

##### <優先順一覧>

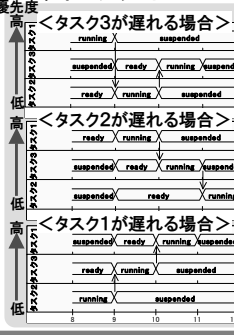
タスク名	優先度	周期	理由
タスク1	1	4ms	制御部は必ず4ms毎に実施する
タスク2	3	10ms	少し(数ms)遅れても問題ない
タスク3	2	4ms	計測データを取りこぼしたくない為、制御部と同じ周期で処理する

##### <タスク間のやりとり>



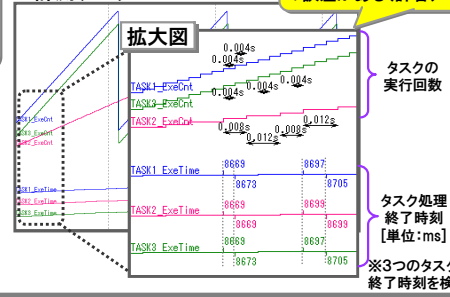
##### 並行性の検証 ※タスクが重なって遅れが発生した時に問題がないかを検討

##### <タイムチャート>



- 最悪のケースは、送信が遅れることになるが、各タスクの処理時間(1ms未満)から計測データを取りこぼすことはない。
- 最悪のケースは、新たな要求が上書きされる。⇒仕様として最新の要求が出力できれば良い。
- 最悪のケースは、処理開始が遅れることになるが、各タスクの処理時間(1ms未満)では1周期遅れることはない。

##### <計測データ>

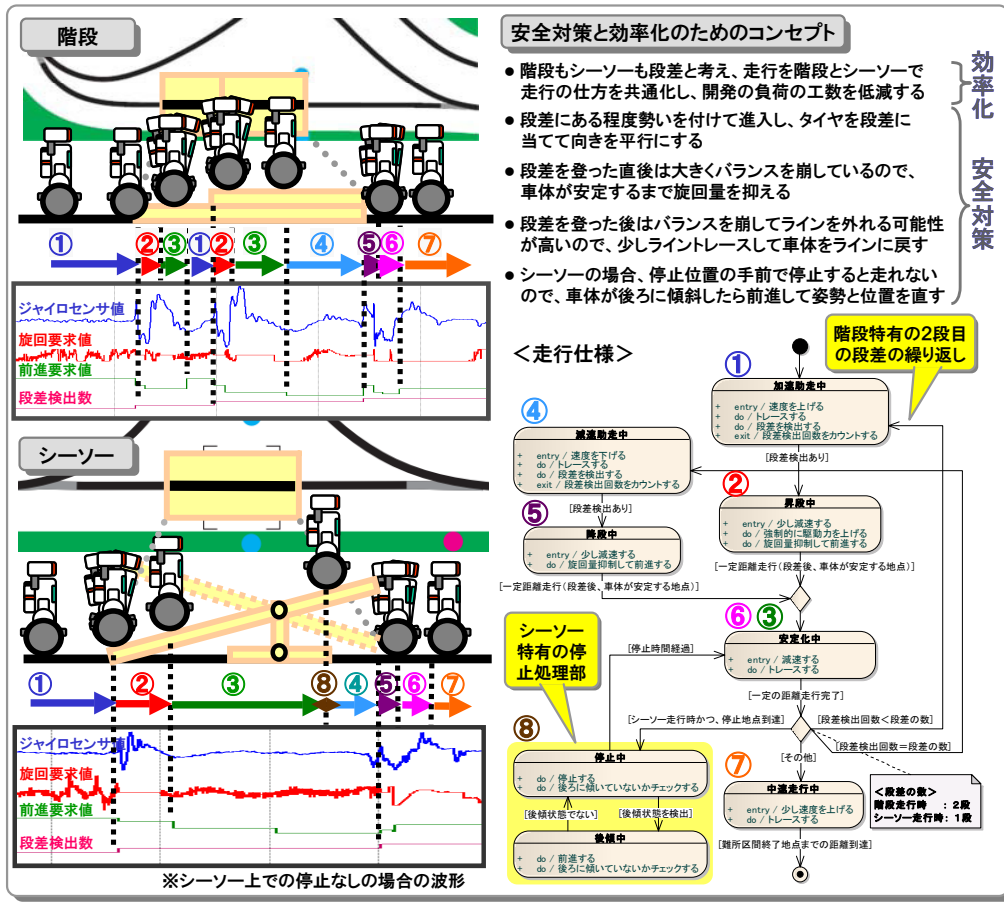


## 4. 性能

## 難所を安全に攻略するための走行戦略と要素技術

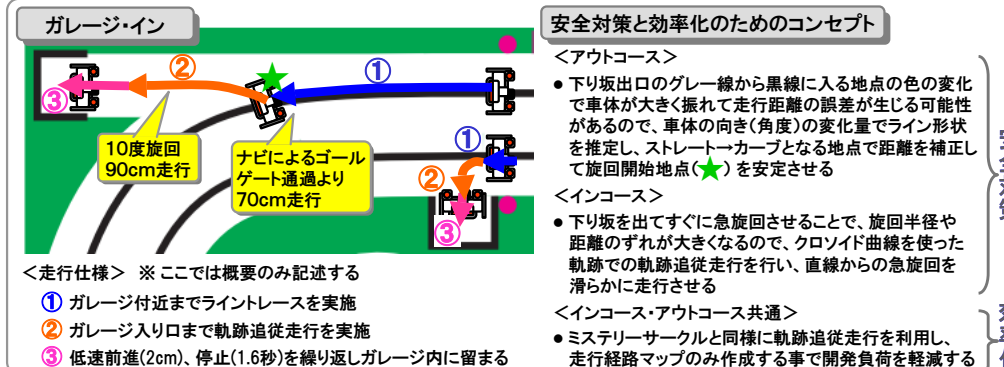
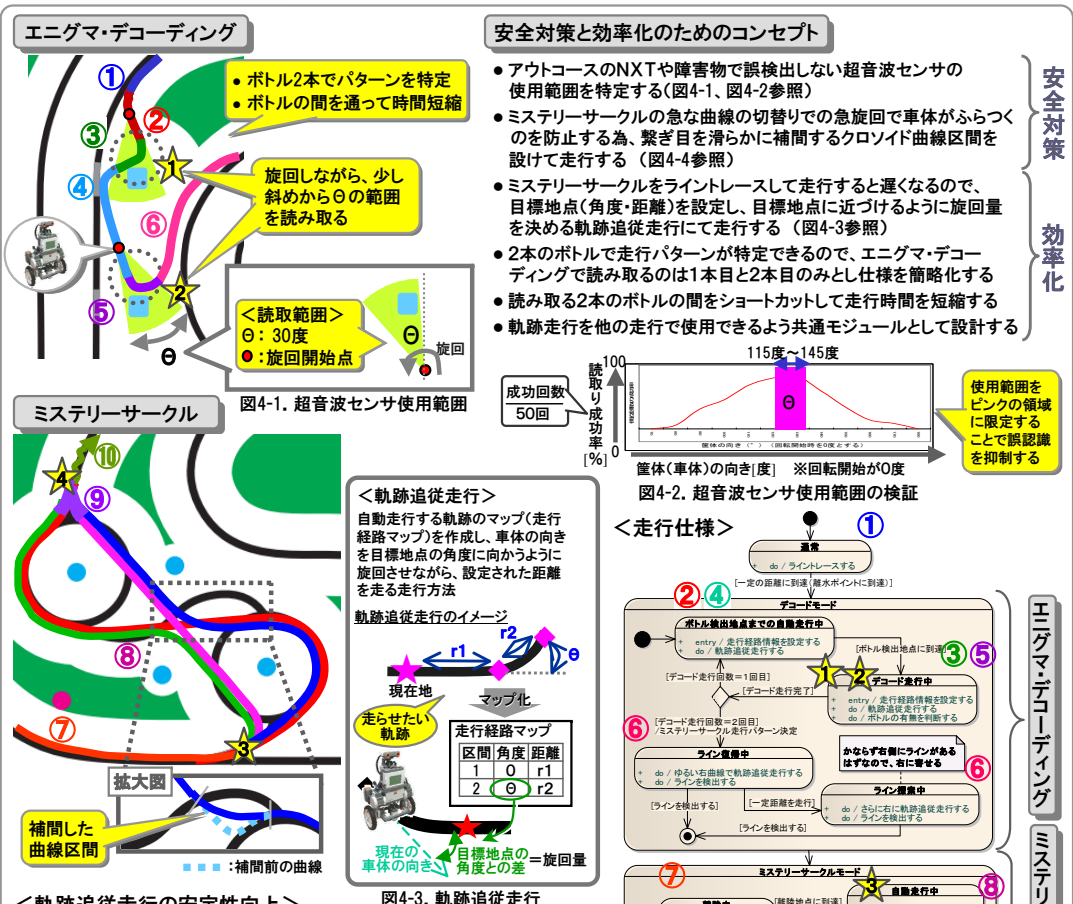
### 〔Ⅰ〕3D難所

階段、シーソーといった3次元(3 Dimension)難所の走行戦略と、それに必要な要素技術を確立して安全に走破する



### 〔Ⅱ〕2D難所

ミステリーサークル、ガレージ・インといった2次元(2 Dimension)難所の走行戦略とそれに必要な要素技術を確立して安全に走破する



### ＜軌跡追従走行の安定性向上＞

#### 課題

走行経路の直線と円弧のつなぎ目で曲率が不連続となり、急激な車輪の加減速が発生し車体がふらつく。

#### 対策

直線と円弧の間に『クロソイド曲線区間(図4-4参照)』を設け、一定の速度・角速度で安定した旋回走行ができる走行経路マップを作成する。(補間部は上記拡大図参照)

※クロソイド曲線は、移動距離に比例して曲率が変化する曲線である。

#### ＜クロソイド曲線の公式＞

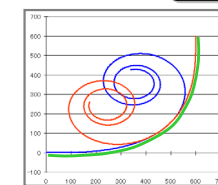
$R$ : 曲率半径、 $L$ : 曲線長

$$R/L = A^2$$

$$L = \pi R/A$$

$$x(l) = \int_0^l \cos \frac{\theta^2}{2} d\theta, \quad y(l) = \int_0^l \sin \frac{\theta^2}{2} d\theta$$

Excelにて算出



クロソイド曲線上の座標

走行経路マップに設定する数だけサンプリング

各輪の推定移動距離から角度と距離を算出

図4-4. クロソイド曲線の導出

## 5. 性能

## 周回タイムの短縮化／非機能要件の実現／開発環境

### 〔Ⅰ〕自己位置推定&PID制御

「PID制御」と「最適な旋回要求指示」を複合させ、コース全域で安定したライントレースを実現！

### 〔Ⅱ〕構造解析

UMLモデルとソースコードをDSMにより可視化して依存関係を管理することで、ソフトウェアの品質を確保！

#### 安定したライントレースのためのコンセプト

##### 課題

- 直線を滑らかに走れるようにPID制御を調整すると、直線⇄カーブの変移点や曲率が異なるカーブが連続するときに追従できない
- ある曲率のカーブを滑らかに走れるようにPID制御を調整すると、曲率の異なるカーブや直線で車体がふらつく

##### 解決方法

カーブに入ったらPID制御の基準値を旋回半径に合った基準値に補正して、コース全体を滑らかにトレースする！

#### 自己位置推定(ナビ)

##### <機能概要>

コース全体をカーブの旋回半径が変わる単位で細かく区切り、車体の向きと移動距離から位置(走行中の区間)を特定する。また、走行中の区間の情報(形状・旋回半径)を管理する。

※ 昨年の20分割から60分割に変更してより細かく制御量の切替ができるようになり、ナビ機能も大幅にパワーアップ

##### <推定方法>

- モータの回転角度から移動距離と車体の向きを把握
- 区間情報と車体の向き・移動距離を照合して走行位置を推定する ※ 区間の区切りは図5-1を参照

#### PID制御の基準値の補正方法

##### <機能概要>

カーブ走行時に、光センサ値の走行閾値との差によるPID制御の基準値を走行中の区間の旋回半径から算出した旋回量(最適旋回量)に補正することで、カーブの走行を滑らかにする。(図5-2参照)

##### <補正方法>

- モータ回転角度から移動距離、車体旋回角度を求め、式5-1から走行区間の旋回半径上の軌跡を取る旋回量(最適旋回量)を求める(図5-3参照)
- 旋回量(最適旋回量)を求める際に必要となるモータ駆動力は、倒立制御で調整されるため、最終的に出力するモータ駆動力をフィードバックさせる(図5-4参照)
- 光センサ値の走行閾値との差から算出したPID制御量に旋回半径から算出した旋回量(最適旋回量)を加算する

試走会での検証の結果、コースの摩擦の違いによるズレは、PID制御で吸収できる範囲内⇒60分割したコース情報の適合は不要

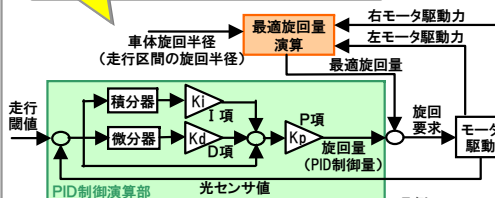


図5-4. 旋回要求指示ブロック図

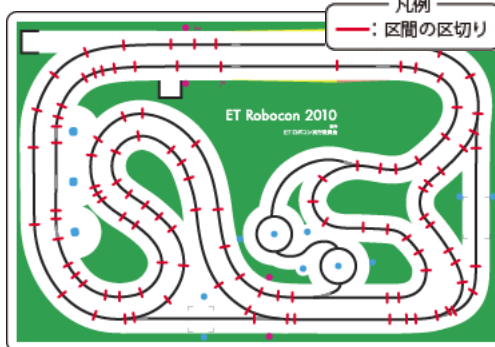


図5-1. 区間分割(ナビ情報)

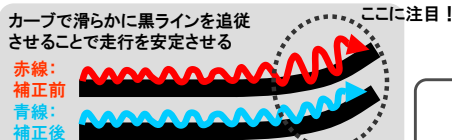
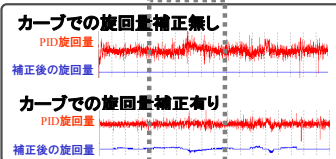


図5-2. ライントレースのイメージ

##### <検証結果>

アウトコースを1周走行時のPID旋回量の波形



補正有りの場合、低周波成分が少なく一定範囲に収まっている⇒安定した旋回量で走行できている



拡大図

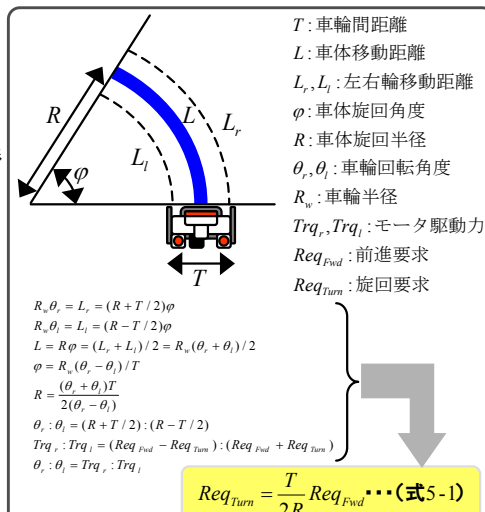


図5-3. 自己位置推定式

#### ソフトウェアの設計構造(アーキテクチャ)解析

非機能要件の一つとなるアーキテクチャ設計について、ソフトウェア品質特性(ISO/IEC9126)のうち、「保守性」「移植性」を維持するための設計ルールが守られているかを市販ツール(Lattix/Understand)により確認。ソフトウェアの関連が複雑化するような変更をしていないか検査し、防止することで上記の品質を維持。

#### モデルの解析結果

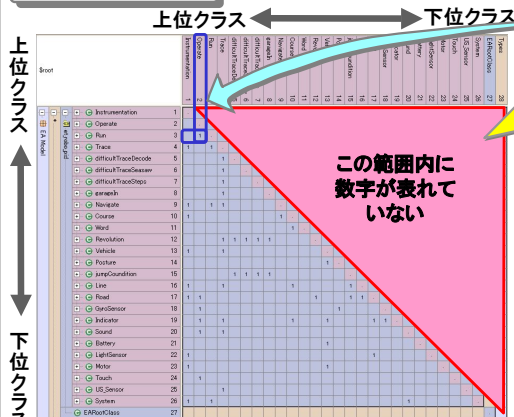


図5-5. 設計初期のLattixツール解析結果

#### ◇図5-4の読み方

Run(3)クラスはOperate(2)クラスに参照されている

下位クラスから上位クラスへの関連がないことから、上位クラスと下位クラスの層別が上手く設計できていることや、双方向の関連が存在しないこともわかる

⇒ 保守性や移植性の向上に寄与

#### ソースコードの解析結果

モデルの解析結果と比較することで、ソースコードで品質を落としていないかが検証できる

⇒ 保守性や移植性の維持に寄与

アクセスの仕方など、設計ルールに違反する箇所のチェックができる

⇒ 保守性や移植性の維持に寄与

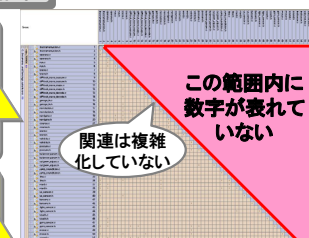


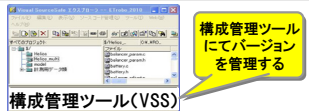
図5-6. 実装後のLattixツール解析結果

### 〔Ⅲ〕開発環境

ソフトウェアの安全設計のために、開発成果物を管理し、ミスを抑制するためのルールと特性や性能を検証するための計測環境を整備。

#### 成果物管理

モデル  
ソースコード  
登録  
単体テスト環境  
のプロジェクト



#### ルール類

コーディング  
ルール  
(命名規約を含む)

MISRA-Cに準じた社内規約により、見易さとコーディングによる不具合を防止する

#### 計測環境

改良版GamePad  
任意のデータを32byte可能計測可能

データをインポート  
計測データの入力により、修正ソフトを再現テストする

データ解析ツール  
モータ特性や制御量の変化を検証

PCデバッグ環境(Visual C++)