2025年度未踏IT人材発掘・育成事業

ソフトウェア設計のコード化と数学的検証を実現する Goフレームワークの開発 — Design as Codeと新しい仕様駆動開発の実現 —

戸田朋花（株式会社サイバーエージェント・株式会社AbemaTV）

Design as Code を実現するライブラリ goat を開発

goatとは

Go 言語を用いて設計を記述し、モデル検査で
設計の正しさを検証できるライブラリ

- Protocol Buffers などの IDL の自動生成
- E2E テストコードの自動生成
- シーケンス図の自動生成

goatのコアバリュー

- コードを書くような設計体験
- 設計の正しさの検証
- 設計と実装の統合

検査できること

- ☑ どの状態でも常に成り立つこと
  （例）在庫システムにおいて在庫数が常に負にならない
- ☑ P が起きたらいつかQが起きること
  （例）リクエストを送ったら必ずレスポンスが返る
- ☑ 最終的に必ずある条件が成立する
  （例）分散システムの各ノードが最終的に一致した状態に収束する
- ☑ 何度でも繰り返し成り立つこと
  （例）送信待ちキューが何度でも空になる

```go
// StateMachineのスキーマを定義
serverSpec := goat.NewStateMachineSpec(&Server{})

// StateMachineに状態を定義
serverSpec.
    DefineStates(serverInit, serverRunning).
    SetInitialState(serverInit)

// Eventのハンドラーを定義
goat.OnEvent(serverSpec, serverRunning,
    func(ctx context.Context, event *request, server *Server) {
        goat.SendTo(ctx, event.Sender(), &response{
            Exists: true,
        })
    },
)

// インスタンスを生成
server, _ := serverSpec.NewInstance()

// ルールを定義
rule := goat.Always(goat.NewCondition("count>0", server, func(s *Server) bool {
    return s.Count > 0
}))

// モデル検査を実行
err := goat.Test(
    goat.WithStateMachines(server),
    goat.WithRules(rule),
)
```

goatによる
仕様記述例

2025年度未踏IT人材発掘・育成事業

ソフトウェア設計のコード化と数学的検証を実現するGoフレームワークの開発
― Design as Codeと新しい仕様駆動開発の実現 ―

戸田朋花（株式会社サイバーエージェント・株式会社AbemaTV）

要件定義・設計特化の AI Agent である YAGI の開発

YAGIとは

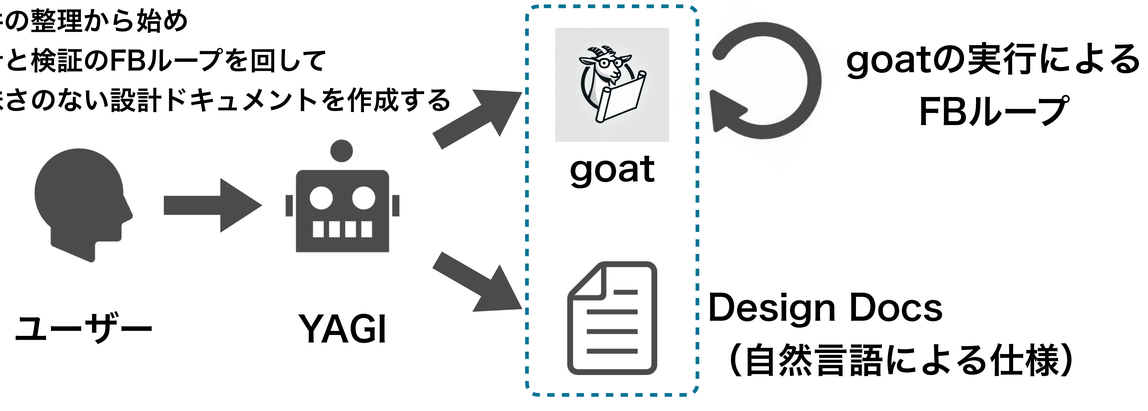
- CLI型のAI Agent
- 設計フェーズでのAI活用支援
- Spec-Driven Development の実現

YAGIの実行例

```
User -> 会社が会議室予約を管理するためのシステムを設計せよ。現在、従業員はメールやスプレッドシートで会議室を予約しており、重複予約や競合が発生している。このため、会議室の空き状況をリアルタイムに確認できる集中管理型の予約プラットフォームを提供する必要がある。システムは重複予約を防止するために時間が重なる予約を禁止し、予約の作成・キャンセル・確認をサポートしなければならない。また、クライアントアプリケーション、サーバ、データベースで構成され、予約情報はサーバメモリではなく必ずデータベースに永続化されなければならない。外部カレンダー連携は範囲外とする。

Agent ->
[🐐 Requirements Phase] Organizing requirements...

  Loading template...
  Appending to design_docs.md...
  Appending to design_docs.md...
  Appending to design_docs.md...

## Requirements Summary

**Background**: 現在の会議室予約システムでは、メールやスプレッドシートの使用により重複予約や競合が発生しています。

**Goal**:
  - 会議室の空き状況をリアルタイムで確認できる集中管理型予約プラットフォームを提供
  - 重複予約を防止し、予約の作成、キャンセル、確認機能をサポート
  - 予約情報をデータベースに永続化

**Non-Goal**:
  - 外部カレンダー連携

Is this understanding correct?


Requirements phase complete. Continue to schema phase?
```

YAGI による設計の流れ

要件の整理から始め
設計と検証のFBループを回して
曖昧さのない設計ドキュメントを作成する



YAGI の成果物を活用した SDD