高品質ノベルゲーム開発基盤の提案

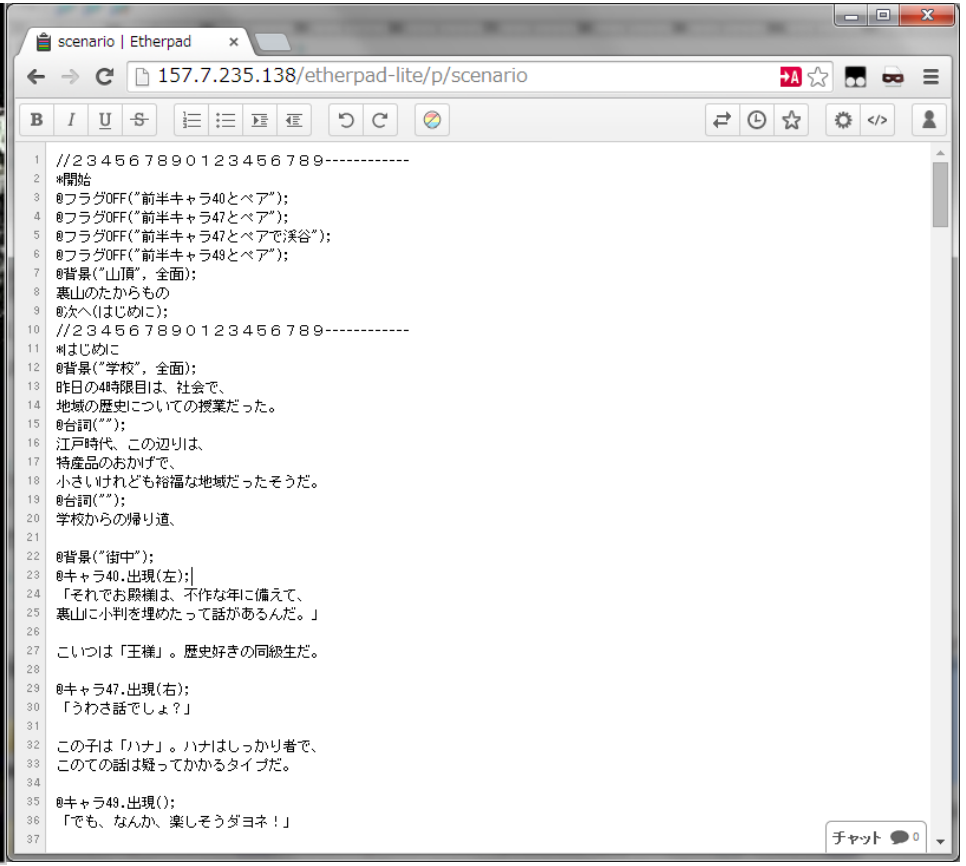
国立情報学研究所 GRACEセンタ/先端ICTセンタ

○ 長久 勝

e-mail：nagaku@nii.ac.jp

デモ

- ノベルゲームの例
- DSLの例
- シナリオの編集



ノベルゲーム開発の現状

- ソフトウェアとしてのアーキテクチャは、1990年代後半に確立されて以降、大きな変化はない
- コンピュータの表現能力の向上に伴い、演出技術は進化を続けている
- 分岐構造の構成技術は、2000年代以降、商品の差別化を図る手法としては高コストと認識されている
- シナリオの分量と、文芸としての品質を担保するため、脚本の共同執筆と同じ手法が主流
- DSLの文法が収斂して10年経つが、IDEなどの支援環境はない

- アーキテクチャが確立されたジャンルのソフトウェア開発においては、通常、ワークフローの整備によって、開発効率や品質の向上を指向するトレンドが生まれるが、ノベルゲームにおいては、残念ながらそうなっていない

ノベルゲーム開発の現状の分析

- 分岐構造の構成技術は、2000年代以降、商品の差別化を図る手法としては高コストと認識されている
  →分岐構造が複雑化すると進行に関する不具合が混入しやすい
- シナリオの分量と、文芸としての品質を担保するため、脚本の共同執筆と同じ手法が主流
  →ゲームを除く文芸的コンテンツでは、分岐を扱わないため、分岐を効果的に用いる手法が確立していない
- DSLの文法が収斂して10年近いが、IDEなどの支援環境はない
  →各自が独自ノウハウに基づいた作業環境を構築している

- アーキテクチャが確立されたジャンルのソフトウェア開発においては、通常、ワークフローの整備によって、開発効率や品質の向上を指向するトレンドが生まれるが、ノベルゲームにおいては、残念ながらそうなっていない
  →ソフトウェア開発ではなく、文芸作品制作の側面が強い

ノベル「ゲーム」の可能性

- 確認：ソフトウェアよりも文芸作品に近いが、
  「分岐」を持つので、文芸作品ではなく「ゲーム」である
- 「分岐」
  ×不具合が混入しやすい、効果的に用いる手法が確立していない
  ○「分岐」を制することで、文芸でありゲームである作品を生み出せる
- ソフトウェア：文芸作品よりも構造化されている
  →作品内の各種情報を計算機で処理しやすい
  - 共同執筆における共有のサポート
  - DSLの変換で香盤表やシナリオフロー（状態遷移図）を自動生成

可能性を信じてやってみた(1)

- GlobalGameJam2013(2013/1/26-27、NII)
  「プログラマは来なくていいです」
  - まずはとっかかり
  - 位置利用ノベルゲーム開発専用エンジン「ARG用アトラスX改」
  - 6チーム18人の開発を観察
    →不具合箇所の特定機能が必要
    →簡単にコードを共有する仕組みが必要
- ニコニコ自作ゲームフェス(2013/3/17投稿)
  「アトラスX改のご紹介-モデル検査もあるよ」
  - 「分岐」を制するために
  - モデル検査機能付ノベルゲームエンジン「アトラスX改」
  - DSLを変換してLTSAでモデル検査
- CEDEC2013(2013/8/21、パシフィコ横浜)
  「モデル検査」のススメ(ゲームシナリオ進行編)
  - 何が検査できるのか整理
  - 聴講者と議論

可能性を信じてやってみた(2)

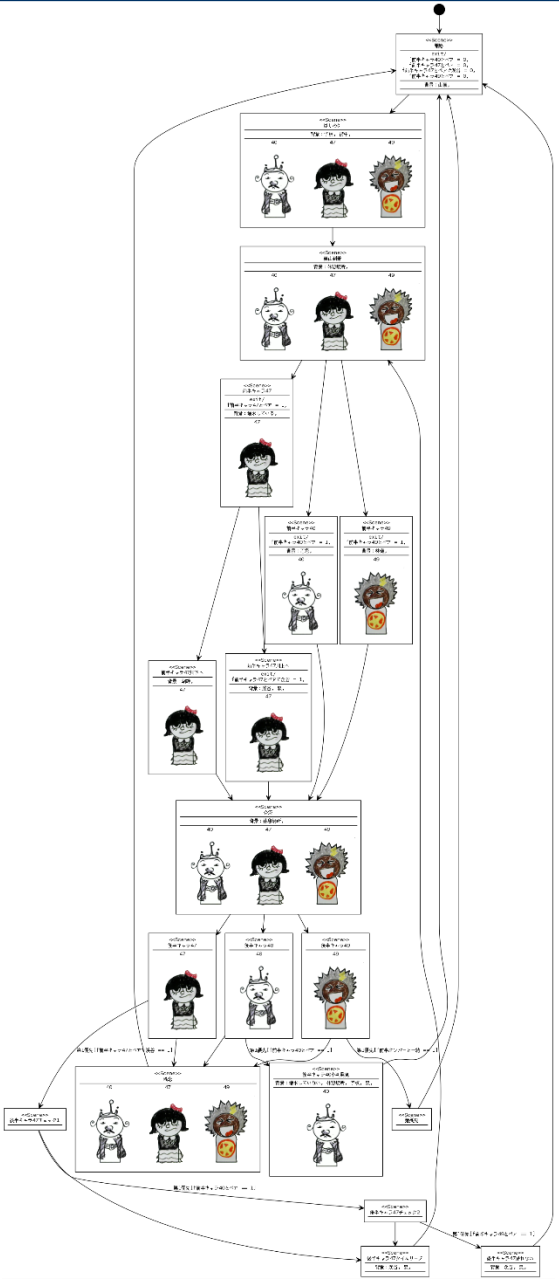
- ゲームとモデル検査ワークショップ#1(2013/9/12、NII)
  - モデル検査機能に注目したワークショップを実施
  - 参加者と議論
- NIIオープンハウス(2014/5/31、NII)
  ワークショップ「ノベルゲームを作ってみよう」
  - GGJ2013で得られたフィードバックを元に機能強化
    →Stypiを用いた共同編集
    →文法check機能＆局所的な文法エラーの影響を受けずに実行可能
    →シナリオフロー
    結果:開発の反復速度が向上した
  - お絵描きワークショップとの統合
- GameCommunitySummit2014(2014/7/5、NII)
  ワークショップ「ノベルゲーJam(ゆるふわ編)」
- XP祭り2014(2014/9/6、早稲田大学)
  ワークショップ「俺の考えたイテレーションの未来!!」
  - Stypiをetherpad-liteに変更
  - 環境をDockerベースに変更

アトラスX改でやってきたこと

- 開発効率向上のための工夫
  - 最新のDSLを簡単に実行し試験可能
    (発表者の実務経験、ソフトウェア工学の知見)
  - DSLの文法定義(発表者の実務経験、ソフトウェア工学の知見)
  - etherpad-liteとの連携により、DSLの共同編集がリアルタイムで可能
    (ワークショップでの実利用からのフィードバック)
  - DSLに局所的な文法エラーが含まれていても実行可能
    (ワークショップでの実利用からのフィードバック)
- 品質向上のための工夫
  - DSLをFSP表現に変換できLTSAを用いたモデル検査が可能
    (ソフトウェア工学の知見)
  - DSLをシナリオフロー図兼香盤表に変換できシナリオ全体を把握しやすい
    (ソフトウェア工学の知見)
  - 数十秒間隔でDSLを解析し、ほぼリアルタイムでシナリオフロー図兼香盤表の更新と文法エラーの把握が可能(ソフトウェア工学の知見)

デモ

- シナリオフロー図兼香盤表の例
- 継続的インテグレーション
  - 文法check機能
  - 存在しないシーンを警告
- 局所的な文法エラーの影響を受けずに実行可能

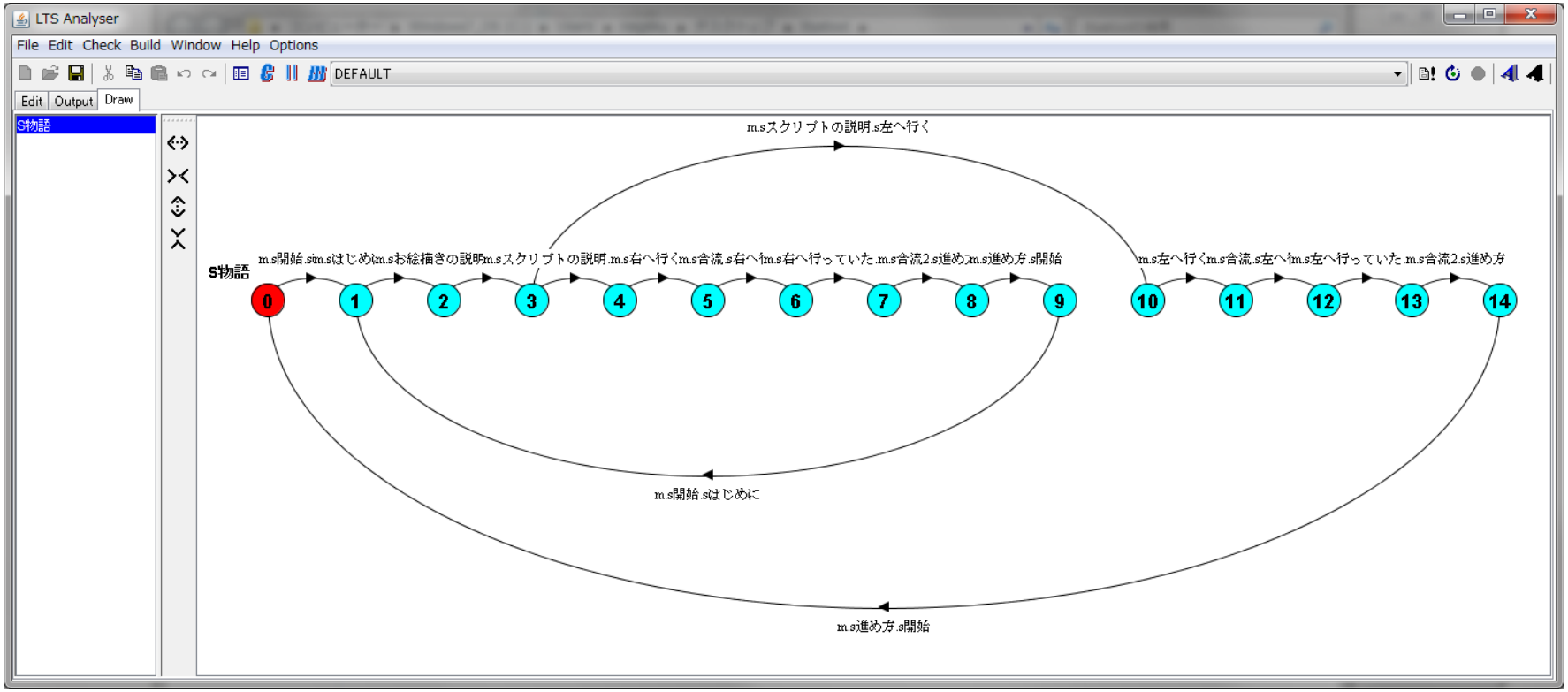


モデル検査

- DSLからモデル記述を自動生成
  - 物語の進行を状態遷移と捉え、FSPによるモデル記述を自動生成
  - 検査項目は人間が書く
- 仕様＞実装＞ユースケース記述
  - 実装は仕様の範囲内
  - 実装はユースケース記述を含む
- 挟み込みで検査式を考える
  - 全称：実装＜仕様
    必ず「魔王が倒される→姫が帰って来る」
  - 存在：実装＞ユースケース記述
    「ライバルが魔王を倒す→姫が帰って来る」場合がある
  - モデル記述の要素で直接的に表現可能な検査は等価性検証が使える
    property Game = (魔王が倒される -> 姫が帰って来る -> Game).

デモ

- LTSAによるモデル検査の例



やってみたら分かったこと(1)

- コードの共同所有
  - リポジトリ:開発者がローカル環境でリソースの編集を行う際には、たいてい1人であり、編集中の状態まで共同所有してはいない
  - ペアプログラミング:物理的に同じ場所にいないと機能しない。3人以上だと成り立たない
  - etherpad-lite:(ワークショップレベルだが)共同編集がリアルタイムで可能
- イテレーションの速度
  - シナリオの分割単位であるシーン毎に文法エラーを評価し、エラーを検出したシーンをダミーに置き換えることで、ゲームそのものは実行可能とした
  - 作業中のシーンに含まれるバグの影響を限定し、複数で同時作業していても、試行錯誤の速度が落ちない
  - 最新のDSLによる実行確認も、ゲームのリロードで可能であり、試行錯誤の速度が落ちない
- 継続的インテグレーション
  - 数十秒間隔でDSLを解析し、ほぼリアルタイムでシナリオフロー図の更新と文法エラーの検出を行っている
  - シナリオフロー図をワークショップ会場にプロジェクションするなど、変化していく状態遷移図の状況を共有することで、進捗や課題を簡単に共有できる

やってみたら分かったこと(2)

- 香盤表
  - 個々のシーンに登場するキャラクタや使用されている背景の情報を合わせて表示している。この情報は、映像作品や舞台演劇で進行管理に使われる香盤表に相当する
  - この情報を元に、音声収録のスケジュールを立てたり、死んだはずのキャラクタが登場しないかモデル検査するなどが可能となる
- 実制作ワークフローとの整合性
  - 商業作品の開発では、全体の構成を設計した後、個々のシーンを分業で書くことが多いが、本事例でも、設計としてシーンと遷移を書いた後、個々のシーンの中身を分業で書いて仕上げる、といったワークフローを採ることができる。更に、作業の様子はリアルタイムに相互参照可能であり、シナリオフロー図で全体像を共有しながら、複数人での作業を無理なく行える
  - シナリオ作成技法として、印象的なシーンを先に設定し、そのシーンが活かされるように逆算的に全体の構成を設計していく手法があるが、本事例でも適用が可能である
  - 分岐構造を持つシナリオの構成を行うツールとして、Twine(http://twinery.org/)などがあるが、本事例を同じような位置付けで使うことも可能である。また、本事例では最終成果物であるゲームまで作れるが、これをプロトタイピングとすることも可能である

まとめ

- 本事例では、ノベルゲーム開発に、いくつかの技術を導入することで、開発効率と品質の向上について試行した。コードの共同所有やイテレーションの速度、継続的インテグレーション、といった、アジャイル開発のプラクティスを、ソーシャルなWebアプリケーションのように手軽に使える実装として導入し、チームでの開発効率を改善した。また、モデル検査や状態遷移図といった、ソフトウェア工学のツールを導入し、品質の向上も目指した。モデル検査については、手軽にというレベルには達していないが、状態遷移図による現状把握、認識共有は効果的である。他にも、実務改善に使うためのアイデアなどが得られた
- チーム開発を、SNSなどネットワーク上のソーシャルな活動と同じ次元に位置付けることで、実務にも活かせる先端的な体験が手軽に得られると、本事例の取り組みを通じて感じている。今後は、ワークショップなどで手軽に試せる本事例と並行して、より実務志向のツールを中心としたワークフローを提示したい