CWI Centrum Wiskunde & Informatica | IC3D MEDIA | Hogeschool van Amsterdam Amsterdam University of Applied Sciences | CREATE-IT APPLIED RESEARCH

Early Quality Assurance in Software Production

Taaltechnologie voor betere Game Software

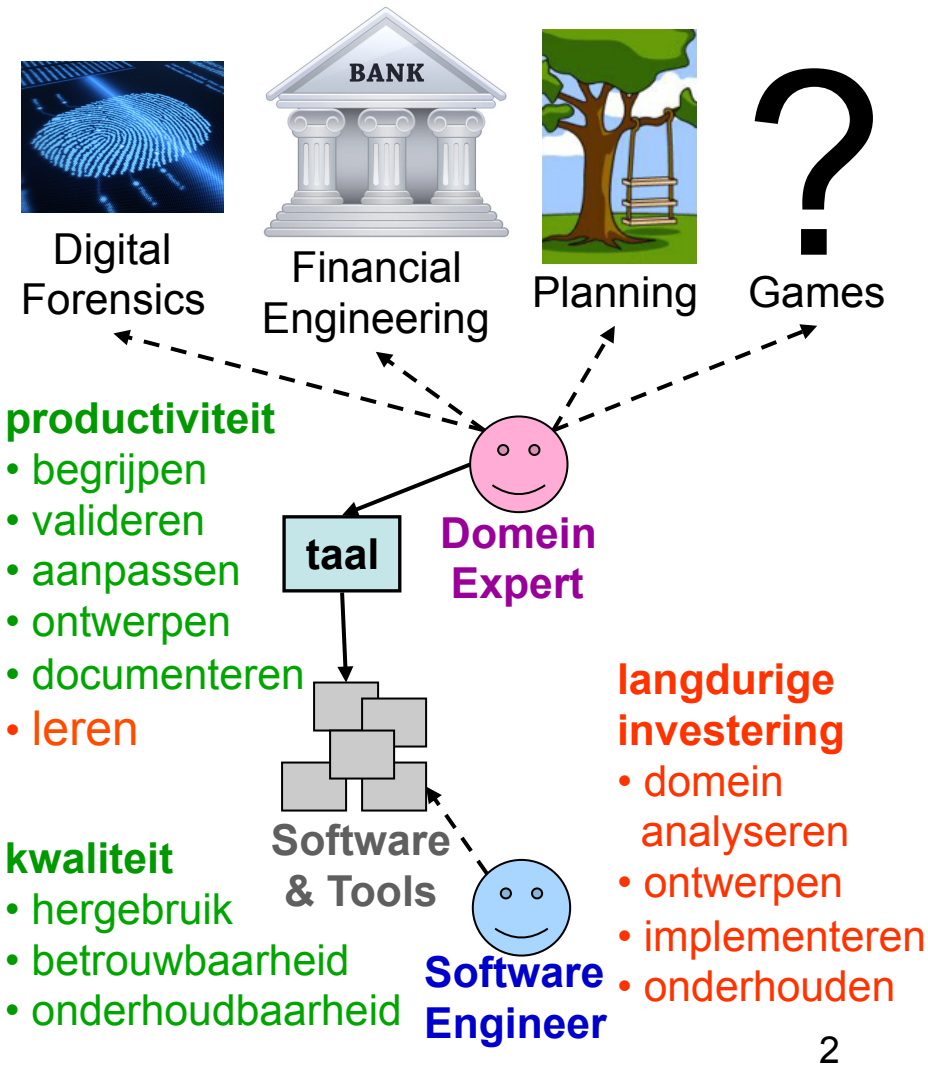
EQuA Symposium
27 Juni 2013

Riemer van Rozen (rozen@cwi.nl)
HvA / Create-IT applied research &
CWI / Software Analysis and Transformation (SWAT) group

In samenwerking met
Joris Dormans (HvA), Paul Klint (CWI),
Loren Roosendaal (IC3D Media), Tijs van der Storm (CWI)

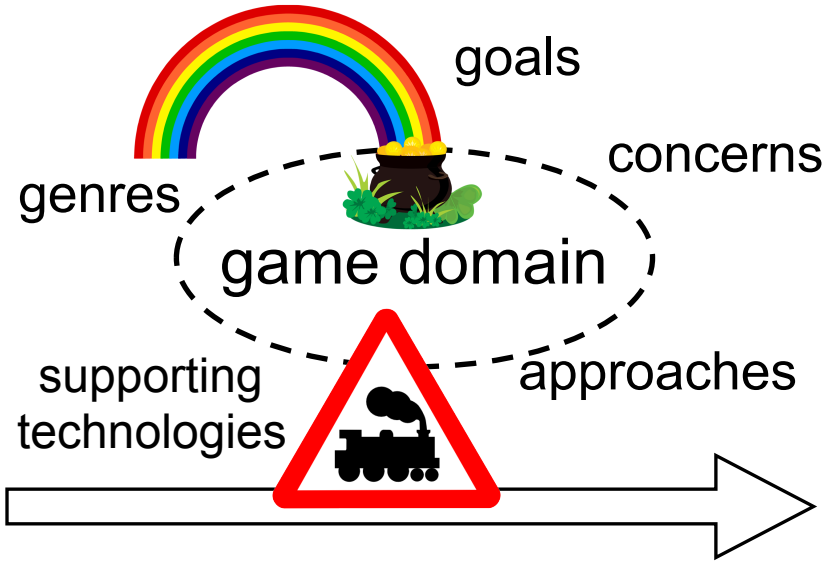
Introductie

- Domein-specifieke taal (DSL)
  - programmeertaal of uitvoerbare specificatie taal
  - biedt passende notaties en abstracties, expressieve kracht gericht op, en meestal beperkt tot, een bepaald probleem domein [Deursen2000]
- Baten
  - verbeteren productiviteit
  - verbeteren kwaliteit
  - verschuiven werk naar expertise
- Kosten
  - langdurig investeren, engineering
  - scholen werknemers
- Kunnen DSLs worden toegepast op het game domein?



Probleem

- Probleem
  - Weinig inzicht in hoe de voordelen van domein-specifieke talen in game development te behalen
- Redenen voor gebrek aan DSLs
  - Game domein is diffuus, breed
  - Ontwikkeling technologieën
    - Domein analyse
      "tracks a moving target"
  - Bestaande benaderingen
    - Game engines
    - Wetenschappelijke benaderingen
  - Snelheid is alles
    - time to market: enorme druk
    - maximale uit technologie
      - performance
      - schaalbaarheid

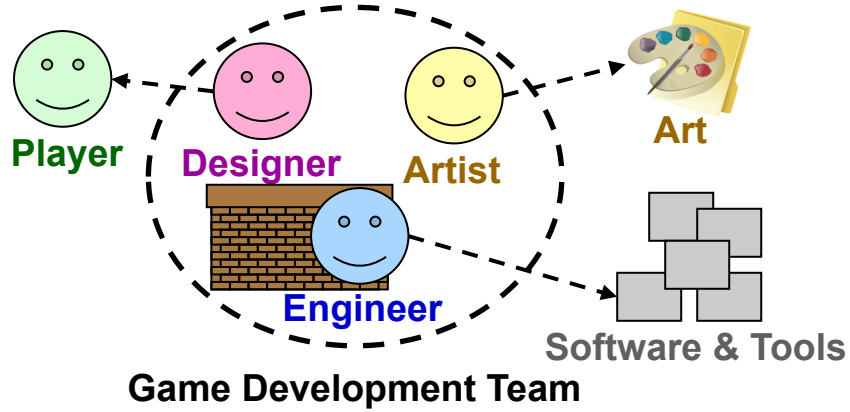


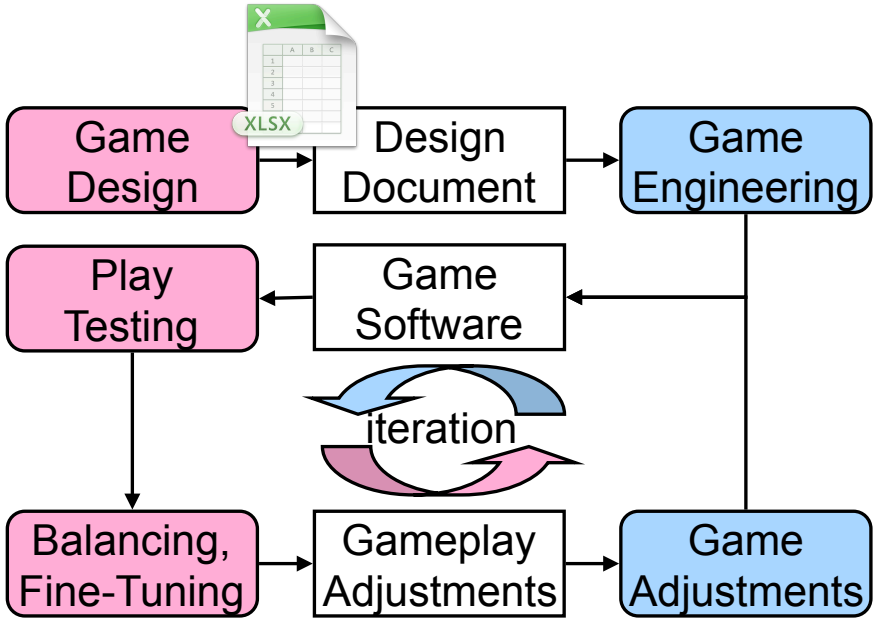
het game domein:
groter dan de filmindustrie,
zeer breed
en constant in beweging

urgent oplossingen nodig

Probleem

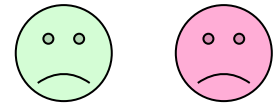
- De kwaliteit van game software staat onder druk
  - Time to market beperkt de tijd voor creëren, implementeren verbeteren van games
- Game ontwikkelingsproces
  - Game Designer
    - designs speler ervaringen
    - creëert een game design document
    - voert play testing uit
    - brengt balans in spelregels
  - Software Engineer
    - ontwerpt en implementeert game software
    - maakt aanpassingen





Probleem

De gameplay balans moet worden aangepast!



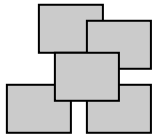
Player Designer

Design
mechanics, rules
dynamics
aesthetics

Gebrek aan middelen voor Fine-tuning

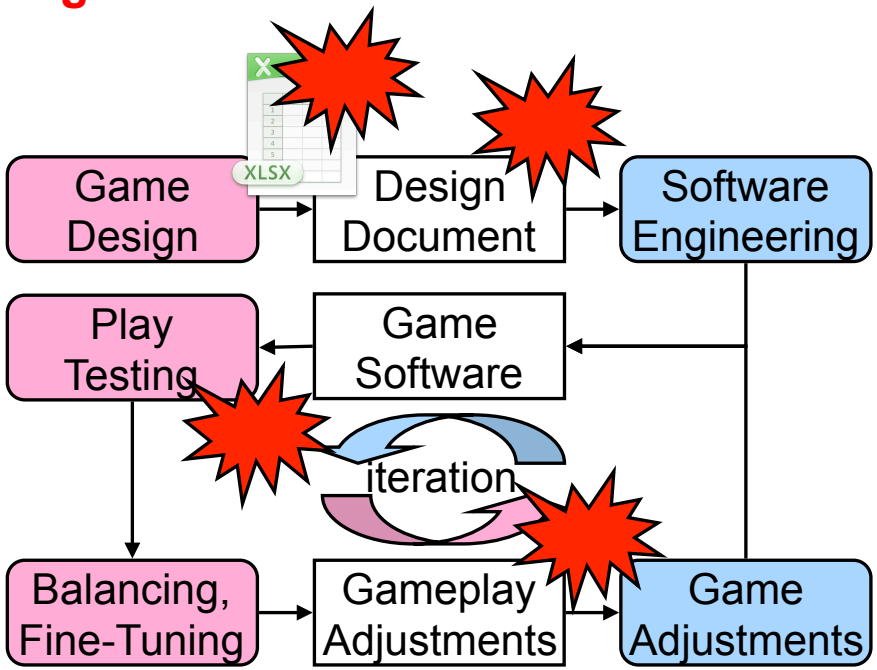
Ok, welke code aanpassingen moet ik daarvoor maken?

Software
libraries, APIs
game engines
tools



Engineer Software

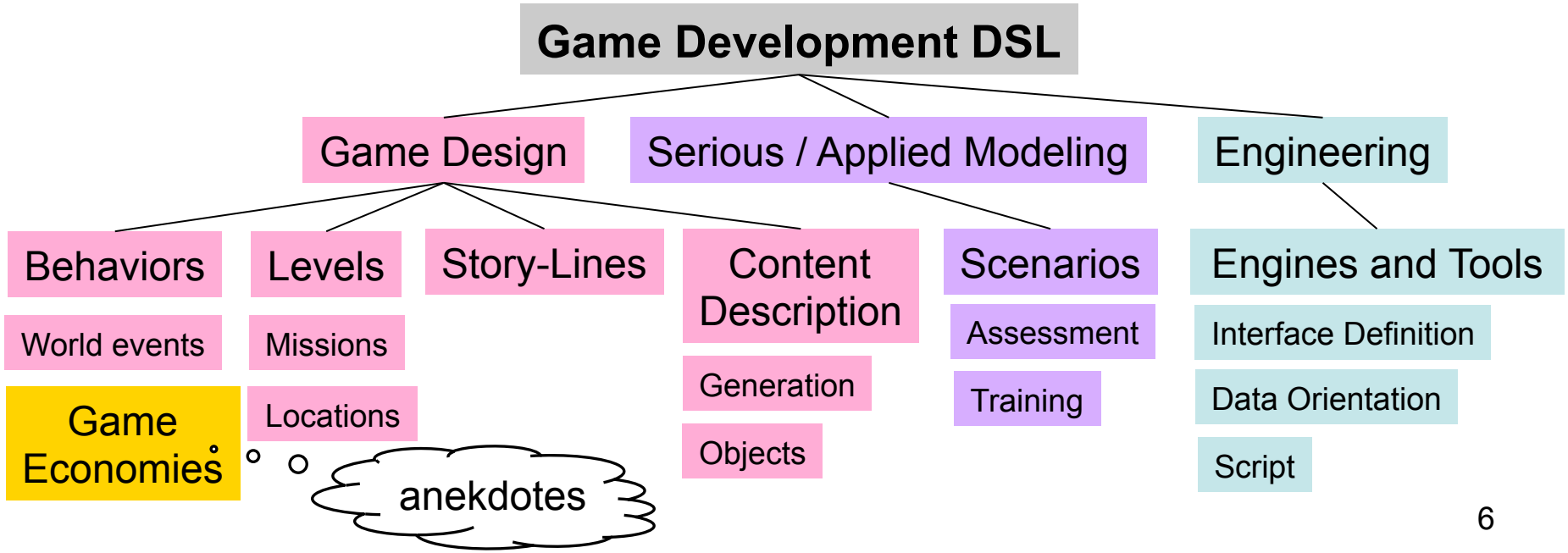
- Tekortkomingen in het proces zijn te herleiden tot gebrek aan
  - gemeenschappelijk vocabulaire om game regels te communiceren
  - samenwerking
    - game design doc → requirements
  - middelen voor fine-tuning
    - play testing → gameplay aanpassingen



EQuA

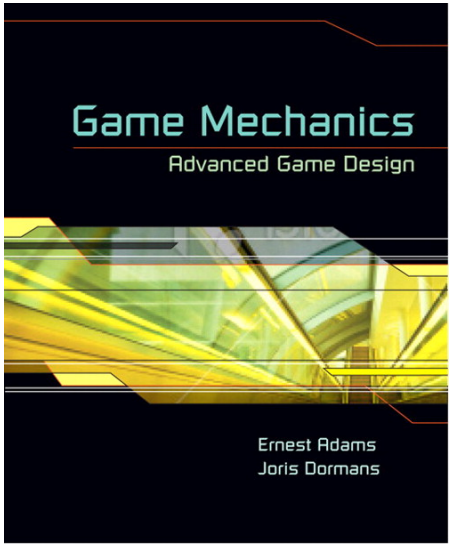
Doelstellingen

- Doelstelling
  - Aantonen dat een game ontwikkelingsproces kan profiteren van domein-specifieke talen
- Mogelijkheden
  - domein-specifieke talen en libraries per concern
    - light-weight, reusable, interoperable, extensible

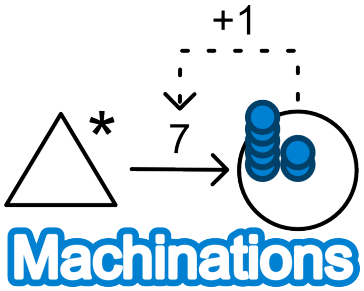


Machinations

- Machinations
  - is een visuele taal voor game designers
  - beschrijft de regels van interne game economieën
  - geeft game designers een gemeenschappelijk vocabulaire
    - creatief redeneren
    - delen
  - plaatst structurele kwaliteiten van game mechanics die bijdragen aan “goede gameplay” op de voorgrond
  - vormt vanuit ons perspectief een domein analyse



[Adams en Dormans]

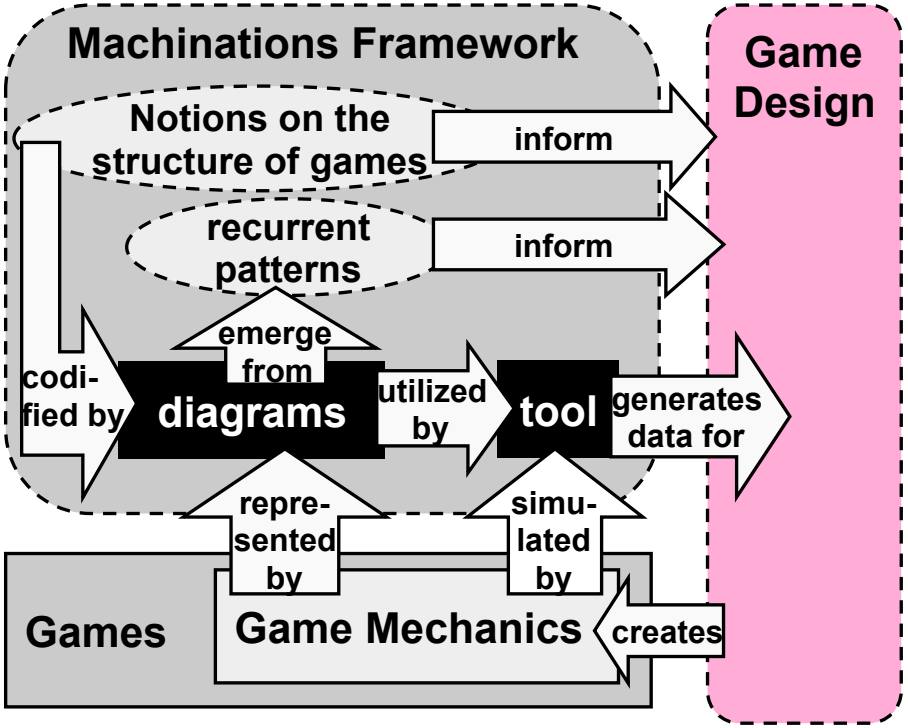


Het Machinations logo bevat een feed-back loop

steps	1	2	3	4
value	7	14	21	28

Machinations

- Machinations
  - beoogt game design hulpmiddel te zijn
  - geeft feedback op het game design proces
    - helpt bij het begrijpen van spelregels en het effect op het spelen
    - vormt een toevoeging op paper prototyping
    - streeft naar diagrammen in plaats van Excel sheets

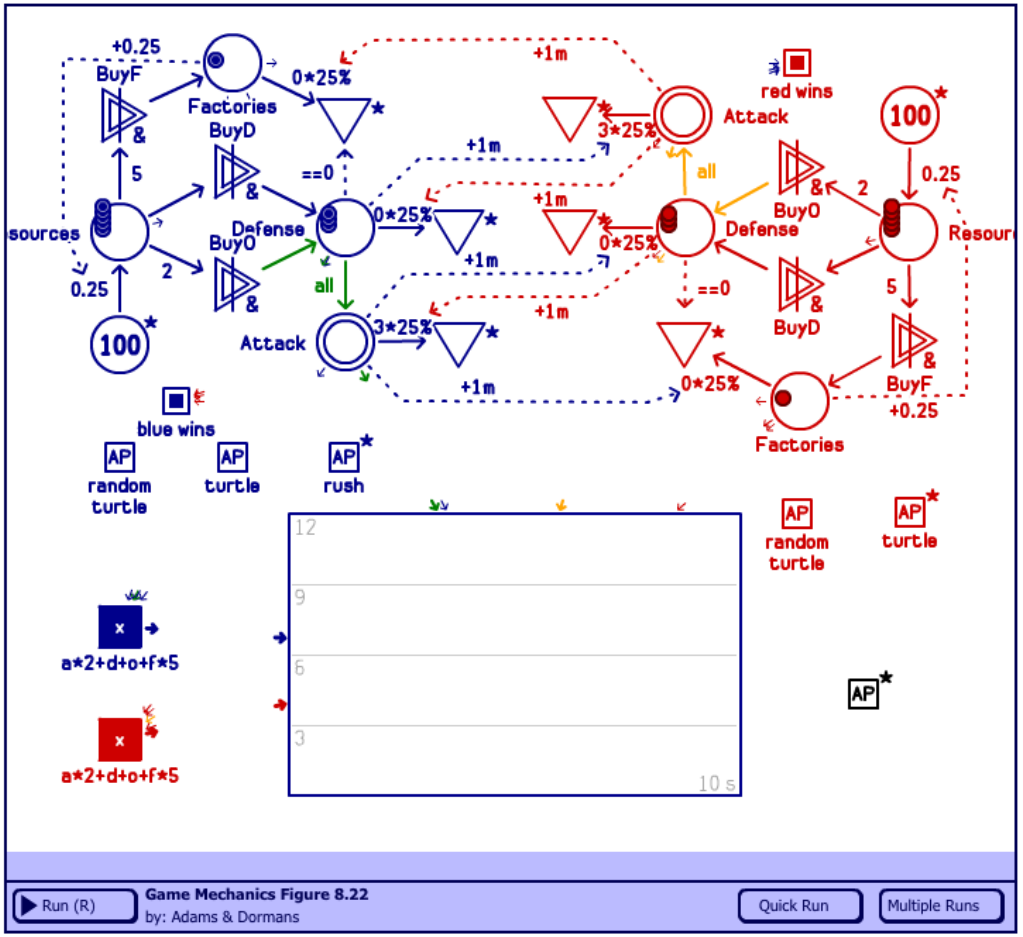


Het originele Machinations framework beoogt het game design proces te ondersteunen [Adams en Dormans]

EQuA

Machinations

- Machinations
  - is een visuele graaf taal
    - nodes
    - edges
  - werkt door herverdeling van resources
    - goud, wapens, kristallen, etc.
  - diagrammen kunnen ontzettend ingewikkeld zijn
    - om te begrijpen
    - om te analyseren



Machinations tool ondersteunt random simulation

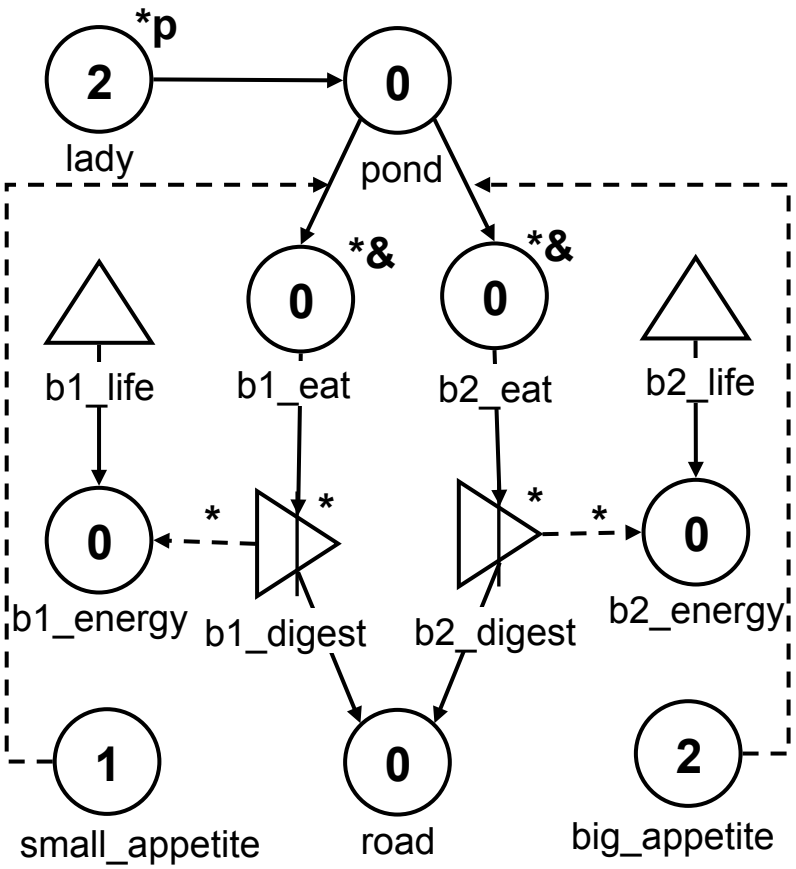
http://www.jorisdormans.nl/machinations/

Micro-Machinations: Introductie

EQuA

Birds Example

- Een vrouwtje voert vogels
  - ze gooit (*p) één voor één (→), twee stukjes brood in een vijver
- Er zitten twee vogels in die vijver. Iedere vogel
  - eet brood (*&)
  - heeft eetlust (→, gestippelde lijn)
    - de ene eet één stukje brood
    - de ander eet er altijd twee
  - verteert brood (*)
    - converter
  - produceert vogelpoep op de weg
  - krijgt energie (gestippelde lijn: *)



Visueel Micro-Machinations model van een vrouwtje dat vogels voert

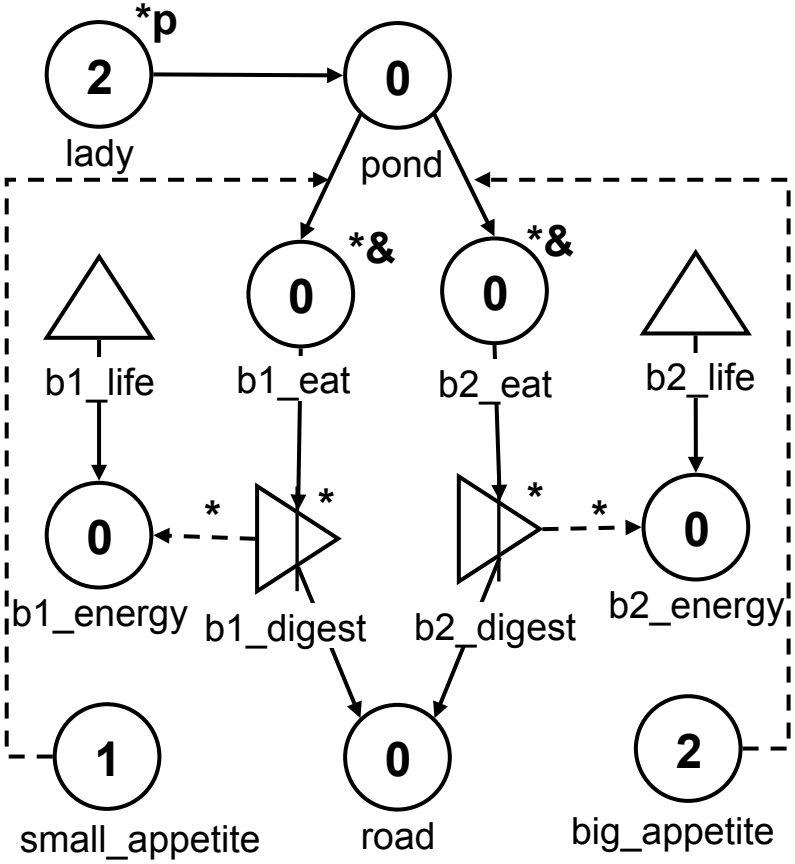
Micro-Machinations: Introductie

EQuA

Game Designer's vragen

- Inspecteren
  - Wat zijn de waarden van pools, welke nodes zijn actief en wat doen ze?

- Selecteren
  - Wat zijn de mogelijke transities voor een state? Zijn er alternatieven?
- Reachability analyseren
  - Gebeurt een taal element in dit model?
  - Is een error state bereikbaar?
  - Kan het spel gewonnen worden?
- Balans verfijnen
  - Zijn de regels goed gebalanceerd?



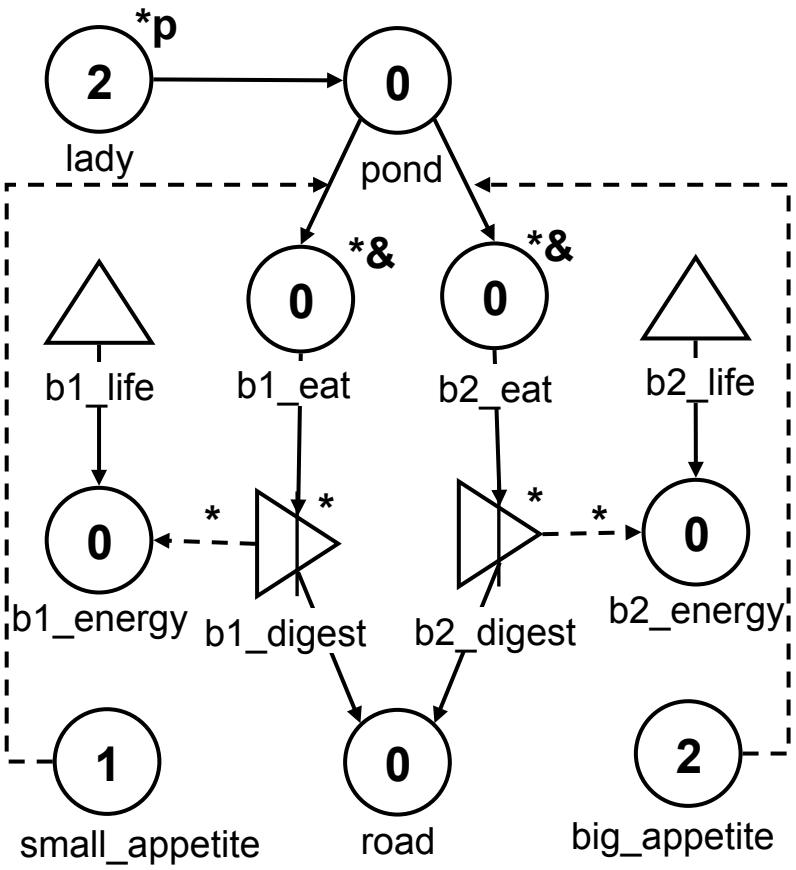
Ondanks de simpliciteit is het lastig om te overzien wat dit model doet

Micro-Machinations: Introductie

EQuA

Technische Uitdagingen

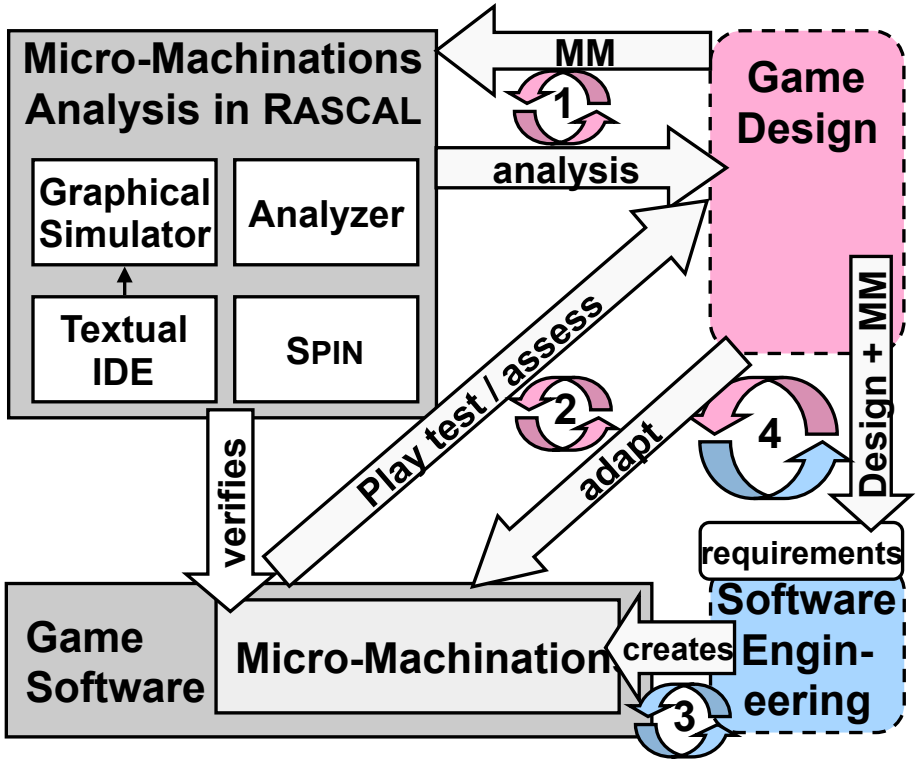
- Parseren
  - tekstuele representatie
- Hergebruik
  - modules
- Inspecteren
  - interactieve omgeving voor visuele inspectie, kiezen van transities en successor states
- Analyseren context constraints
  - statische analyse
- Analyseren reachability
  - model checking
    - Schaalbaarheid
- Prototypes bouwen en aanpassen
  - Light-weight embeddable interpreter met simpele API
    - Interoperabiliteit



Het model demonstreert spiegeling van gedrag, code duplicatie

Micro-Machinations: Doelstellingen

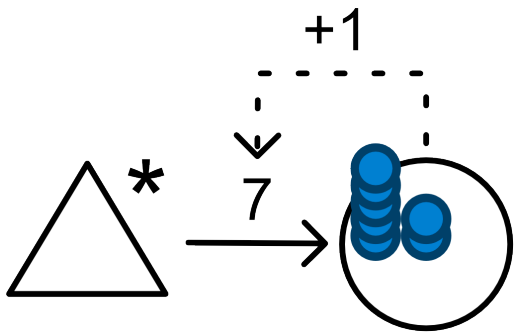
- Introduceren korte onafhankelijke design iteraties (1 en 2)
  - Vroegtijdige kennis over kwaliteit
  - Meer kansen voor aanpassingen en verbeteringen tijdens play testing
- Vrij maken van tijd voor aparte software engineering iteraties (3)
- Reduceren van langere iteraties met onderlinge afhankelijkheden (4)
  - Separation of concerns
  - Model-driven game development



Micro-Machinations beoogt het game ontwikkelingsproces te verbeteren

Micro-Machinations: Aanpak

- Micro-Machinations (MM) is een DSL voor game economieën
  - evolutionaire voortzetting van Machinations die zich richt op
    - software prototyping
    - validatie & verificatie
  - vormt een geformaliseerde subset van Machinations
    - tekstuele taal
    - precieze betekenis
    - reductie van non-determinisme
  - voegt elementen toe
    - modulariteit
    - formele analyse

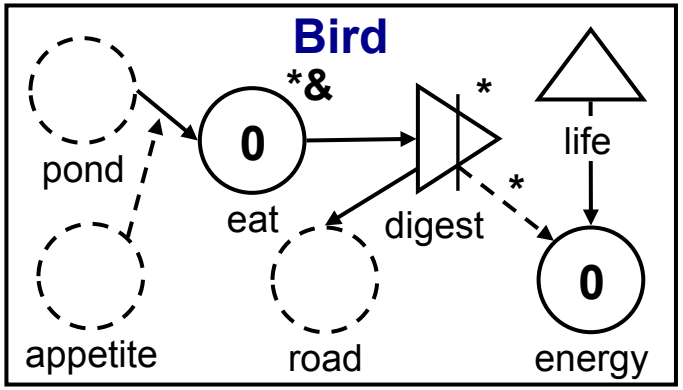


source S **pool** P

S -7*(P+1)-> P

Micro-Machinations: Extensies

- Type definities
  - referenties naar externe nodes (cirkels met stippellijn)
    - Werkt als formal parameters
  - Hergebruik, partiële analyse

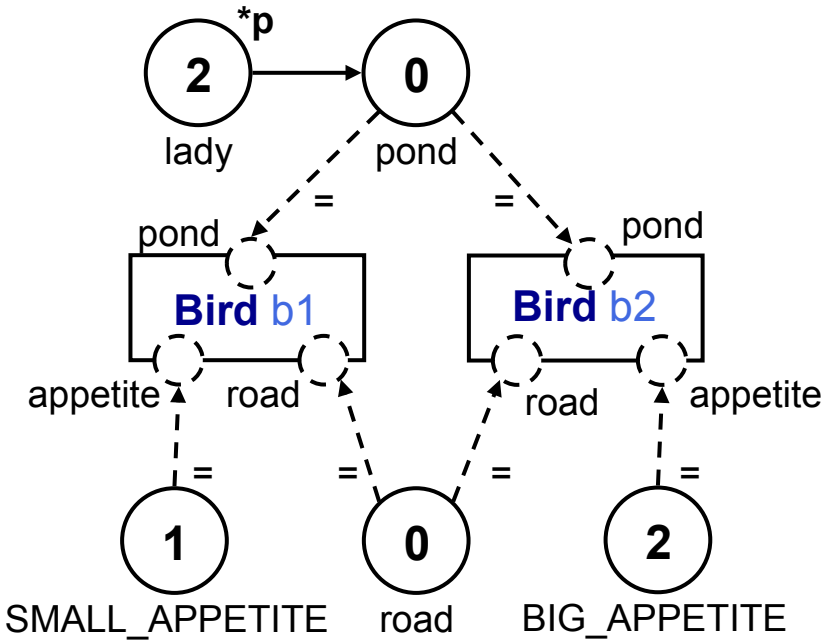



```
//declare Bird type
Bird(ref appetite, ref pond, ref road)
{
  //birds eat all they want (not less)
  auto all pool eat of Bread
  pond -appetite-> eat
  auto converter digest
    from Bread to Droppings
  eat --> digest    //digest 1 Bread
  digest --> road   //produce 1 Dropping
  pool energy of Energy
  source life of Energy        ← !
  digest .*.> energy
  life --> energy

  assert fed : energy > 0 || road < 2
    "birds always get fed"

}
```

- Assertions
  - Naam, Boolean expressie, message
  - Krijgen vogels altijd te eten?

Micro-Machinations: Extensies

- Instanties
  - Referenties worden interfaces
  - Binding interfaces (formal parameters)



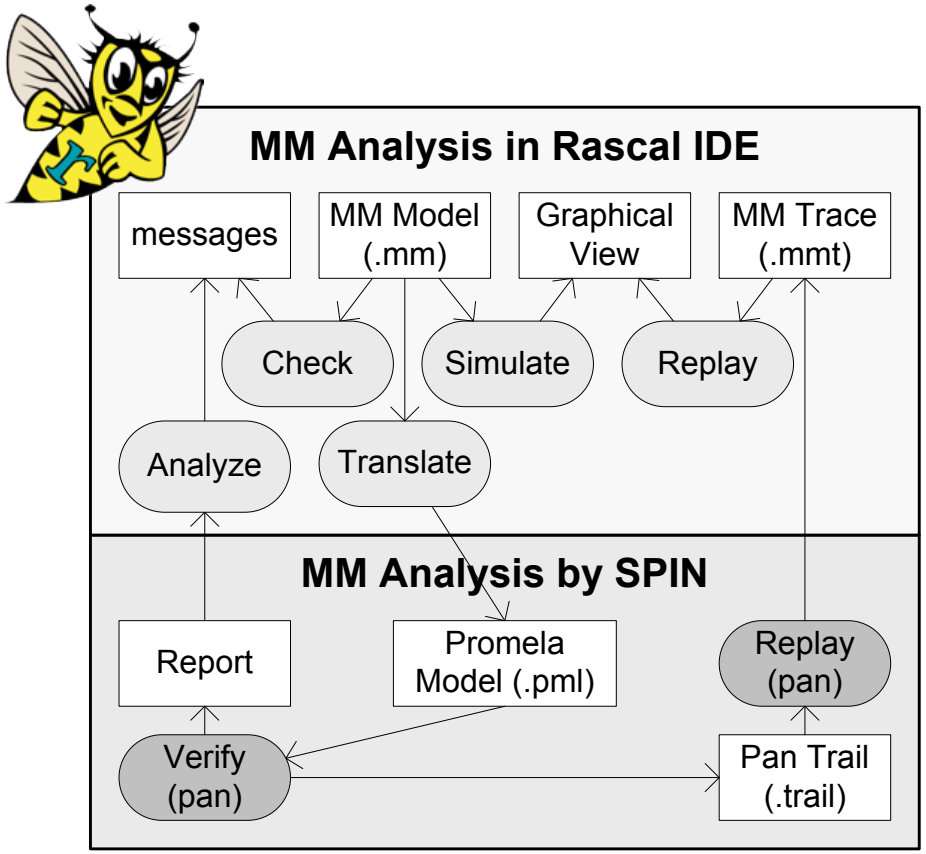

```
unit Bread     : "bread crumbs"
unit Droppings : "bird residue"
unit Energy    : "bird energy“

auto push all pool lady of Bread at 2
pool pond of Bread
pool road of Droppings
lady --> pond

pool SMALL_APPETITE of Bread at 1
pool BIG_APPETITE of Bread at 2

Bird b1
   SMALL_APPETITE .=.> b1.appetite
   pond .=.> b1.pond
   road .=.> b1.road

Bird b2
   BIG_APPETITE .=.> b2.appetite
   pond .=.> b2.pond
   road .=.> b2.road
```

EQuA

Micro-Machinations: Aanpak

- Micro-Machinations Analysis in RASCAL (MM AiR)
  - Workbench voor MM
  - interactieve simulatie
  - visualisatie
  - validatie

- combineert state-of-the-art
  - RASCAL
    - meta-programmeren
    - language workbench
    - rapid prototyping
    - http://www.rascal-mpl.org
  - SPIN
    - model checker
    - schaalbaarheid
    - http://spinroot.com



Micro-Machinations Analysis in Rascal (MM AiR) IDE functies

EQuA

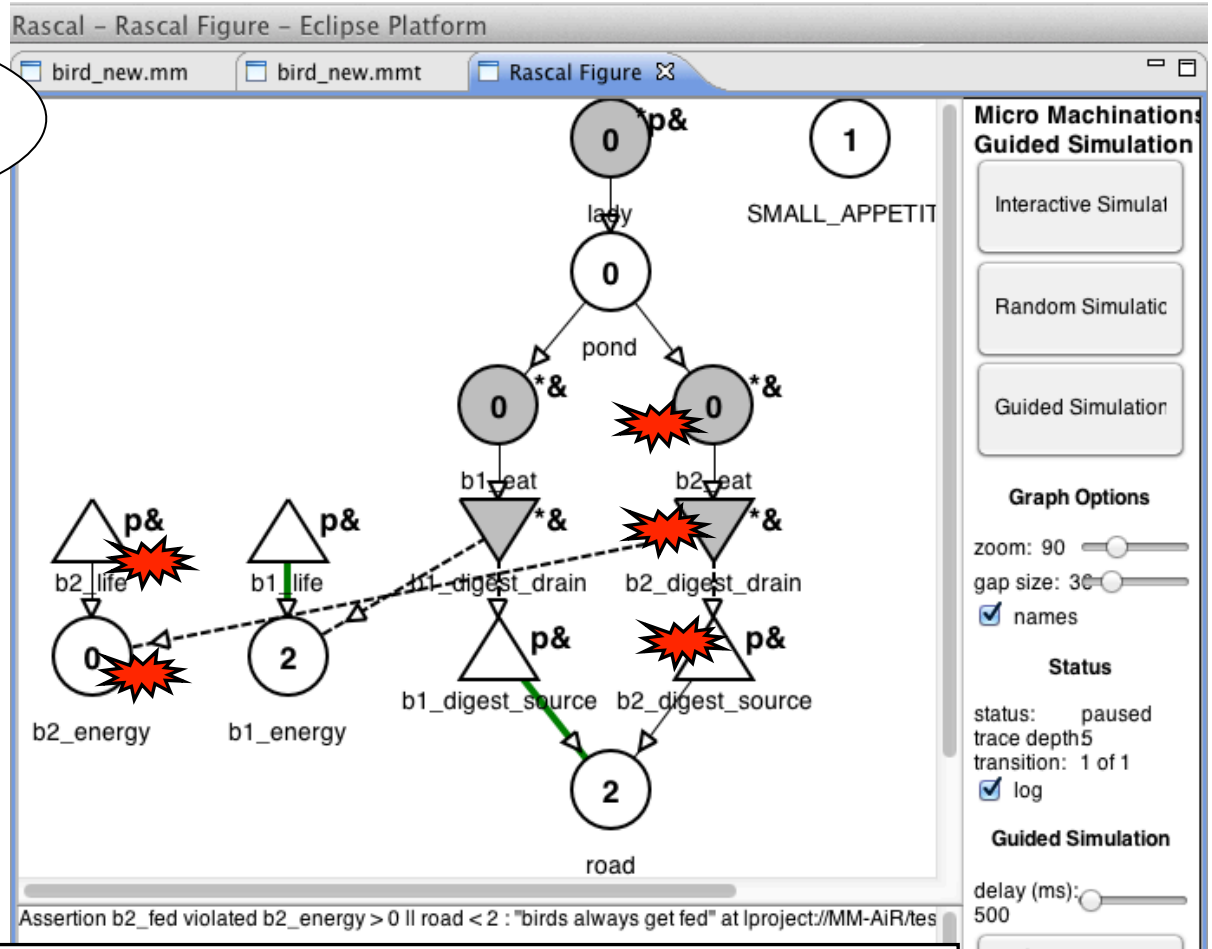
Micro-Machinations: Analyse

Stap voor stap verken ik mijn design, en snap ik hoe het werkt!

Player Designer

```
lady-1->pond
step
pond-1->b1_eat
lady-1->pond
step
pond-1->b1_eat
b1_eat-1->b1_digest_drain
step
b1_eat-1->b1_digest_drain
b1_life-1->b1_energy
b1_digest_source-1->road
step
b1_life-1->b1_energy
b1_digest_source-1->road
step
violate b2_fed
```

Micro-Machinations Trace



Starvation: Node b2_digest_drain does not pull
Inactivity: Node b2_digest_drain does not trigger b2_digest_source
Inactivity: Node b2_digest_drain does not trigger b2_energy
Starvation: Node b2_energy does not pull
Starvation: Node b2_eat does not pull
Starvation: Node b2_digest_source does not push

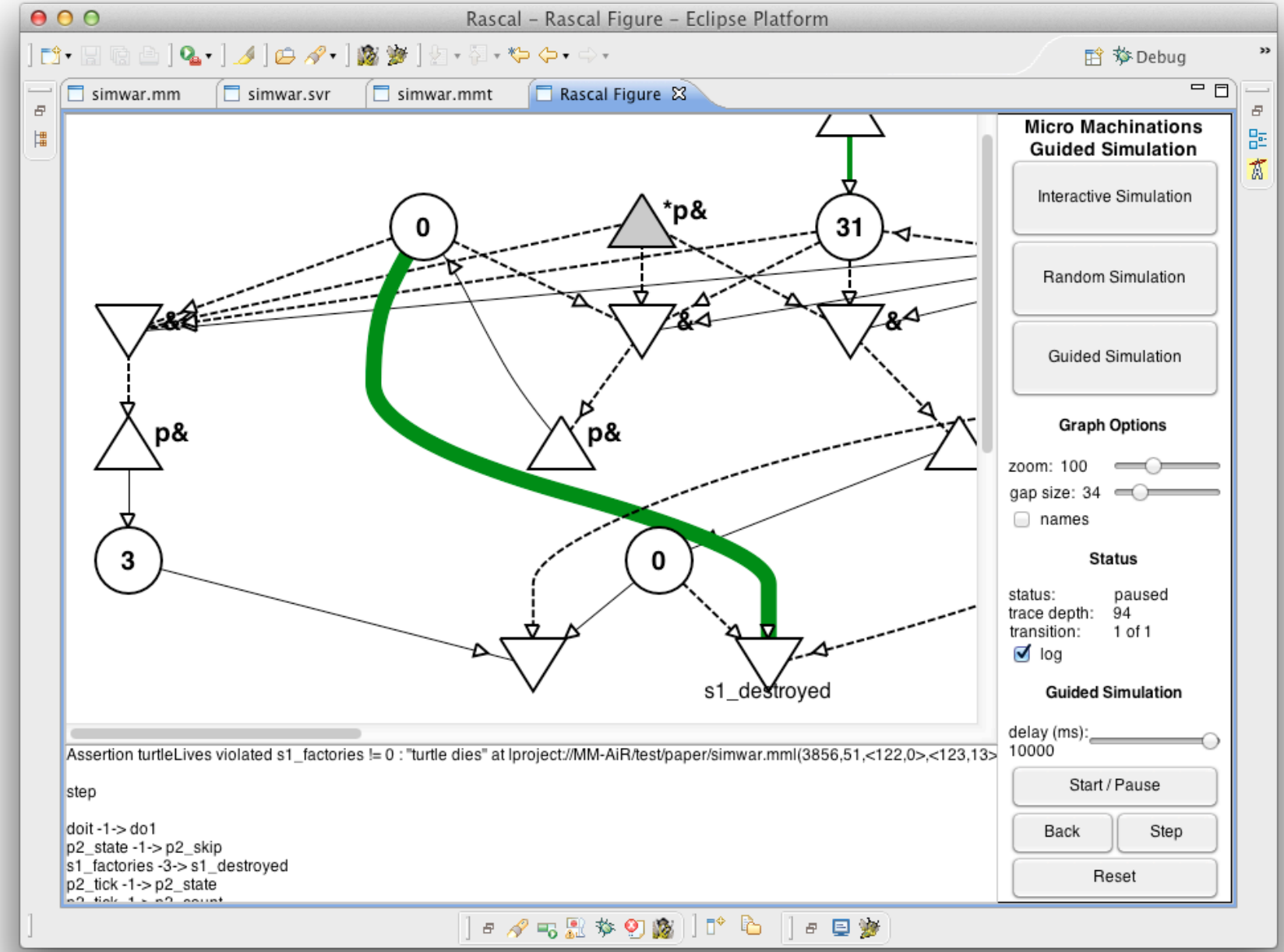
Conclusie

- Een DSL voor game economieën gebruiken geeft inzicht in een spel en de uiteindelijke kwaliteit ervan.

- Lessen
  - Modulariteit is de sleutel voor een game design taal
    - hergebruik van modellen, modulaire validatie
  - Taalfeatures ontwerpen met analyse in gedachten
    - Non-determinisme reduceren → state space reduceren
  - Grafische taal voor designers, tekstuele taal voor tool bouwers
  - PROMELA is een flexibele taal met de benodigde features voor MM analyse
    - Verschillende niet-triviale keuzes leiden tot zeer verschillende schaalbaarheid
  - De RASCAL language workbench is geschikt voor het implementeren MM AiR

- Volgende stappen
  - Micro-Machinations Library

EQuA


Rascal - Rascal Figure - Eclipse Platform
Debug
simwar.mm
simwar.svr
simwar.mmt
Rascal Figure
0
*p&
31
&
&
&
p&
p&
3
0
s1_destroyed
Micro Machinations
Guided Simulation
Interactive Simulation
Random Simulation
Guided Simulation
Graph Options
zoom: 100
gap size: 34
names
Status
status: paused
trace depth: 94
transition: 1 of 1
log
Guided Simulation
delay (ms): 10000
Start / Pause
Back
Step
Reset
Assertion turtleLives violated s1_factories != 0 : "turtle dies" at lproject://MM-AiR/test/paper/simwar.mml(3856,51,<122,0>,<123,13>
step
doit -1-> do1
p2_state -1-> p2_skip
s1_factories -3-> s1_destroyed
p2_tick -1-> p2_state