Proveniência em Jogos

Troy C. Kohwalter

Instituto de Computação – Universidade Federal Fluminense (UFF)

Instituto de Computação

Telemetria de Jogos
Proveniência
PinG
PinGU
Visualização
Considerações Finais

TELEMETRIA DE JOGOS

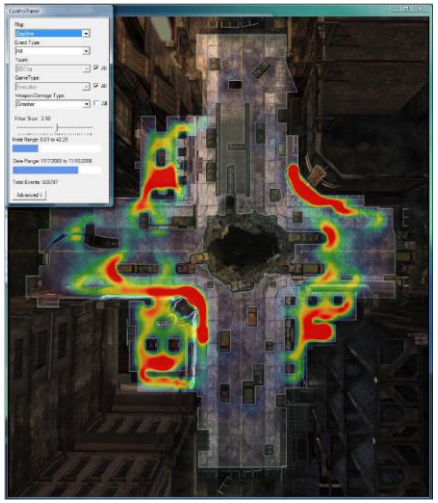
Telemetria de Jogos

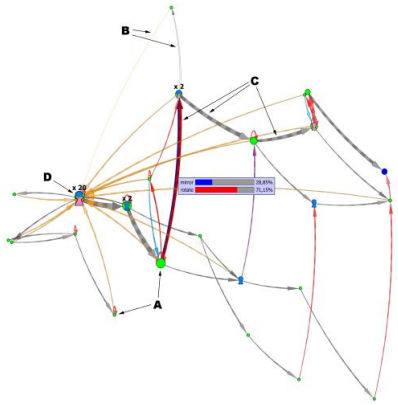
- Coleta de dados
  - Aspecto importante na indústria de jogos
  - Estágio de desenvolvimento e produção

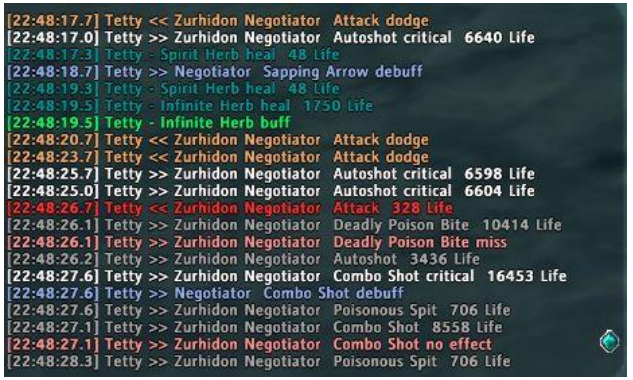
- Vantagens
  - Estabilidade
  - Balanceamento dinâmico
  - Análise de comportamento
  - Monetização
  - Detecção de Problemas

Coleta de Telemetria em Jogos

- Desafiador
  - Complexidade do Jogo
  - Quantidade de informação
- O que coletar?
  - Estados
    - Grão grosso
  - Eventos
    - Grão fino
- Relações de causa-e-efeito
  - i.e., Influencias
  - **Não é capturado!**







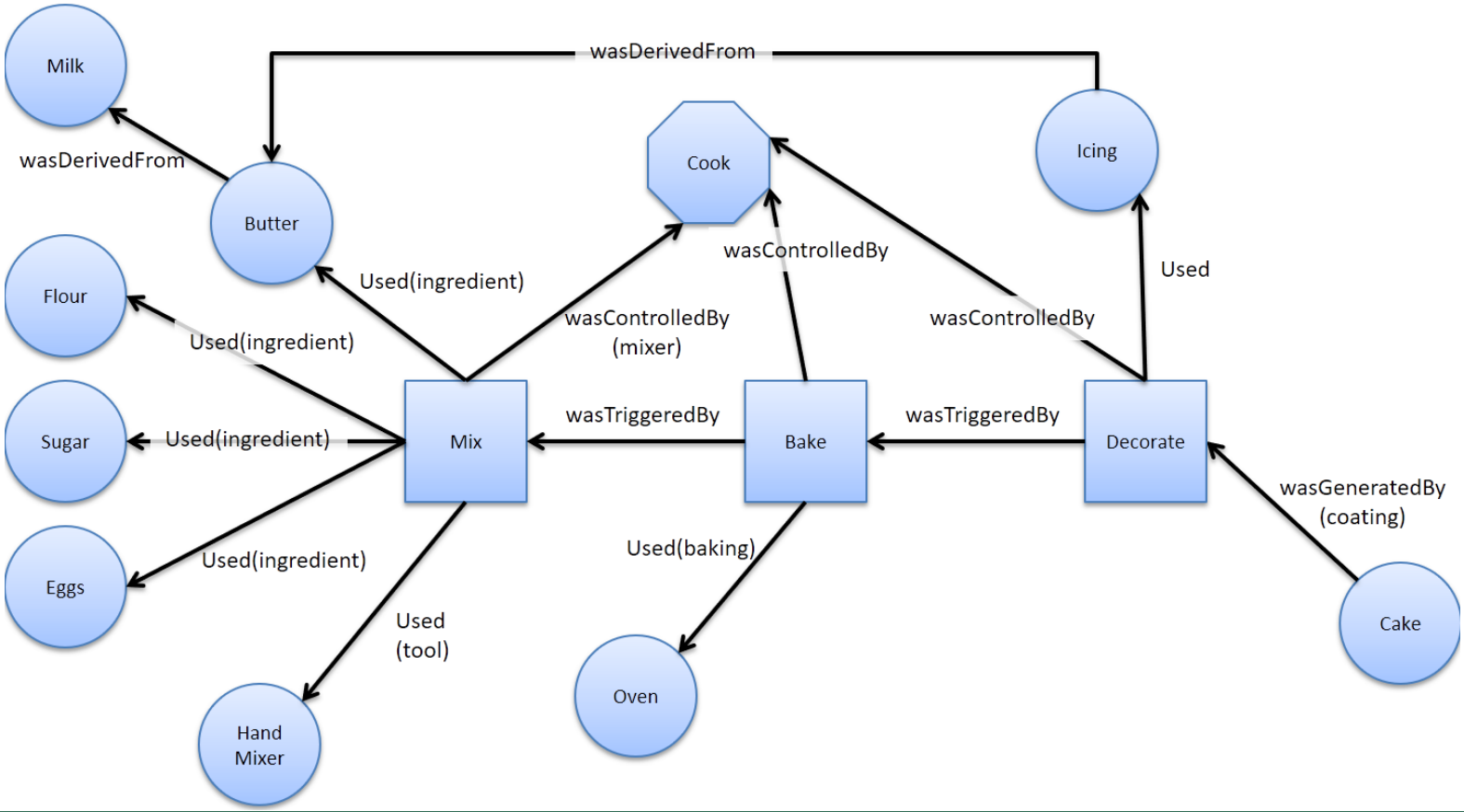
Instituto de Computação

Telemetria de Jogos
Proveniência
PinG
PinGU
Visualização
Considerações Finais

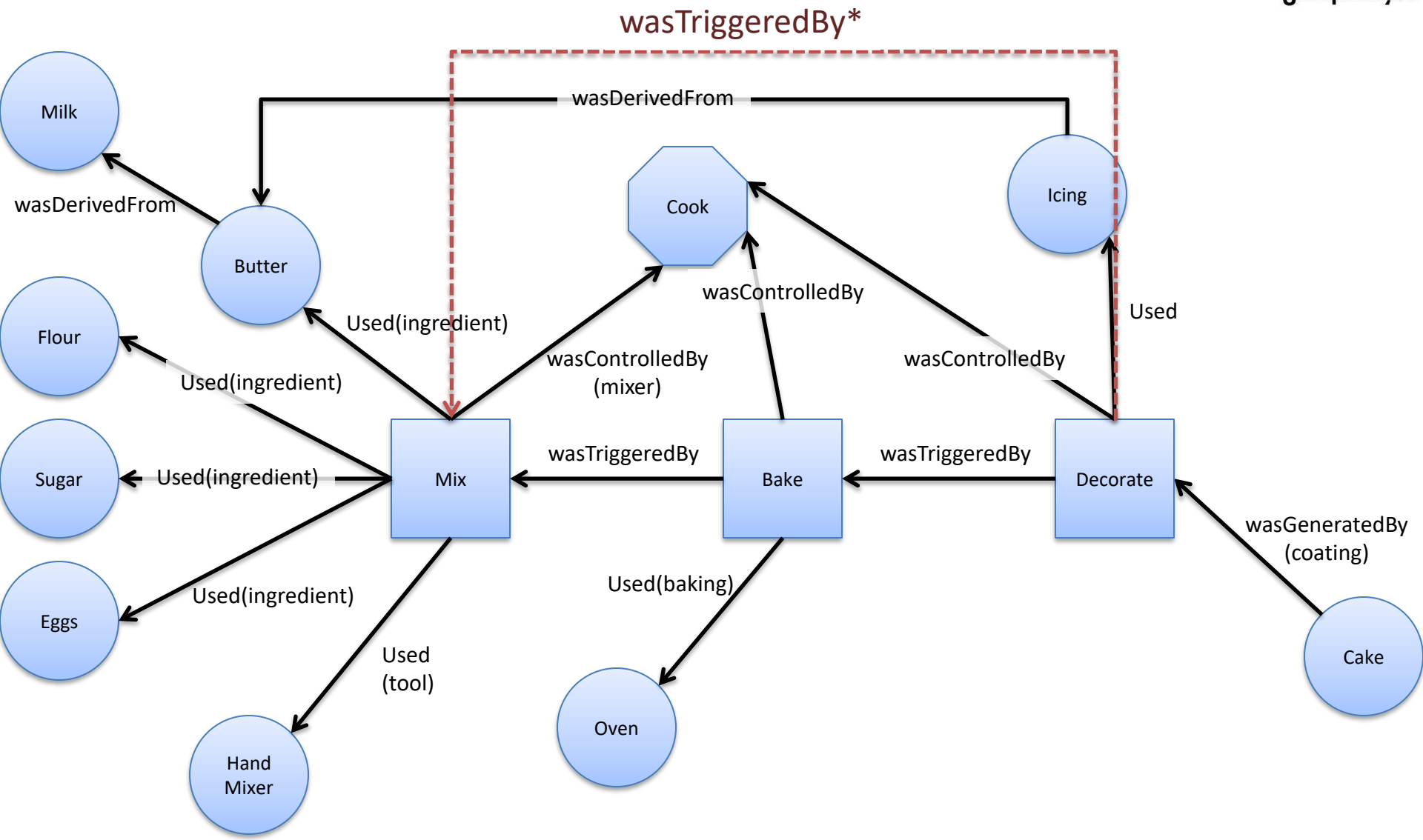
PROVENIÊNCIA

Proveniência

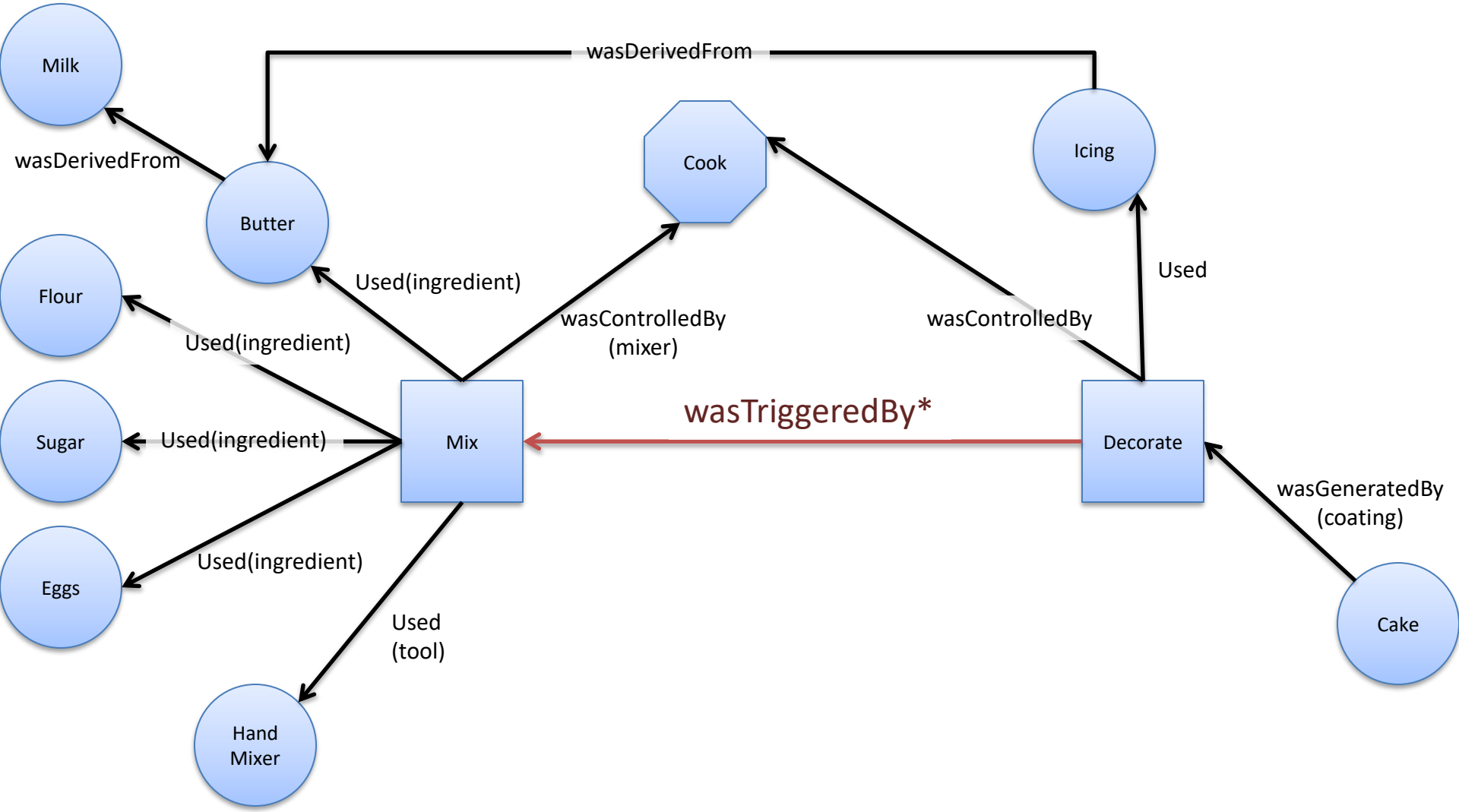
- *"Refere-se à história documentada de um objeto de arte ou à documentação de processos no ciclo de vida de um objeto digital"*



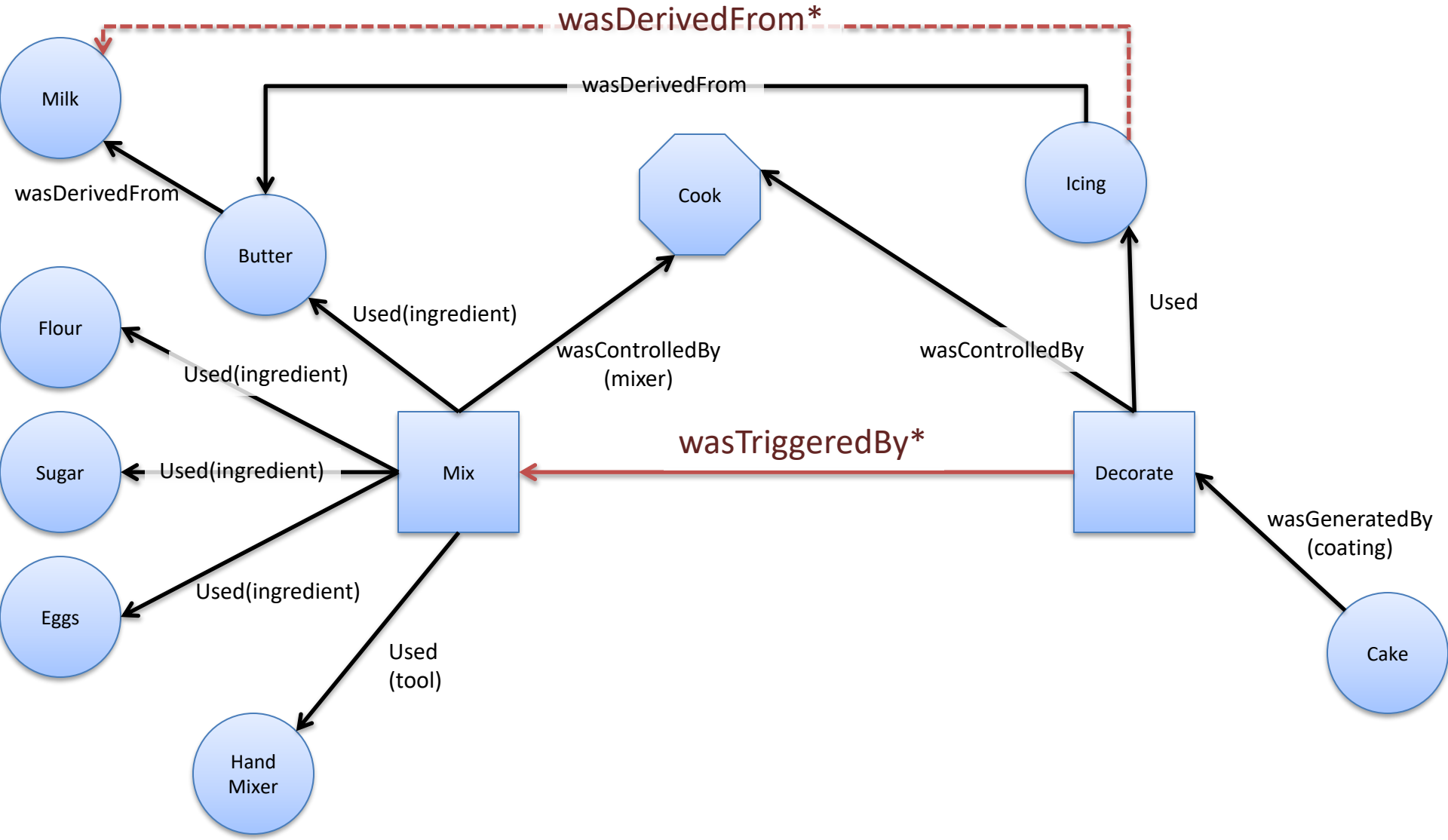
Inferência


wasTriggeredBy*
Milk
wasDerivedFrom
wasDerivedFrom
Butter
Cook
Icing
wasControlledBy
Used
Flour
Used(ingredient)
wasControlledBy
(mixer)
wasControlledBy
Sugar
Used(ingredient)
Mix
wasTriggeredBy
Bake
wasTriggeredBy
Decorate
wasGeneratedBy
(coating)
Eggs
Used(ingredient)
Used(baking)
Used
(tool)
Cake
Oven
Hand
Mixer

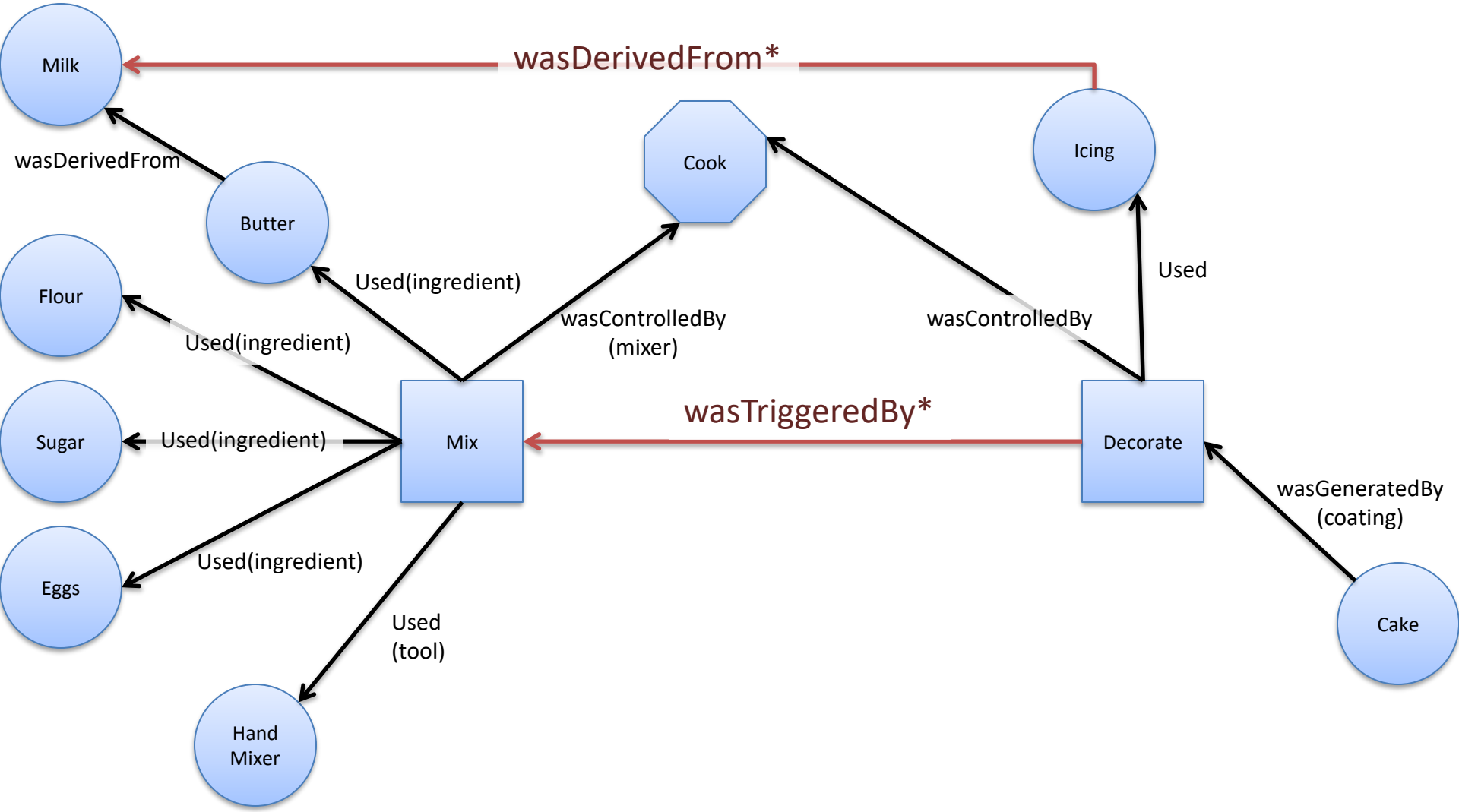
Inferência


Milk
wasDerivedFrom
wasDerivedFrom
Butter
Cook
Icing
Flour
Used(ingredient)
wasControlledBy
(mixer)
wasControlledBy
Used
Used(ingredient)
Sugar
Used(ingredient)
Mix
wasTriggeredBy*
Decorate
wasGeneratedBy
(coating)
Eggs
Used(ingredient)
Used
(tool)
Cake
Hand
Mixer

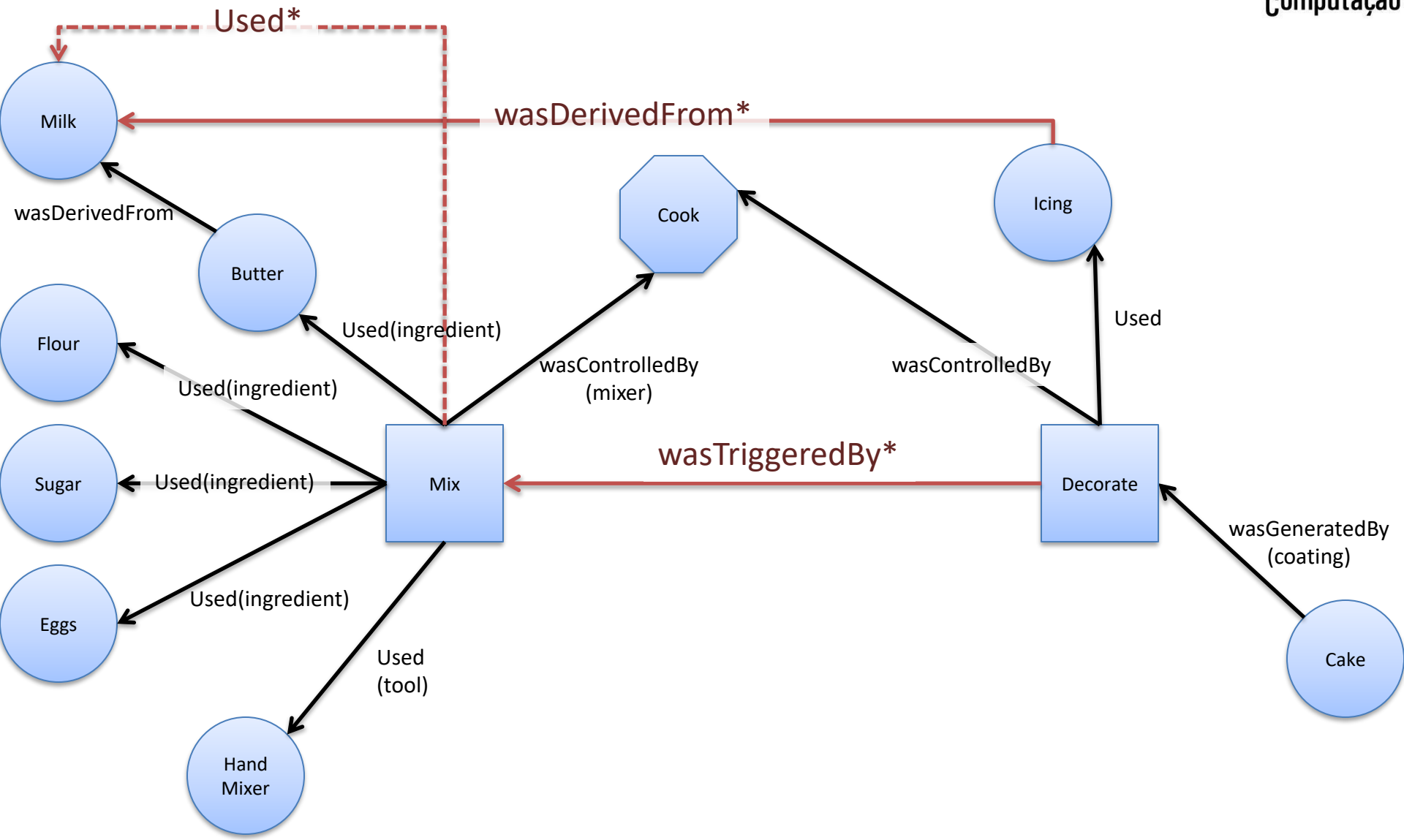
Inferência


wasDerivedFrom*
Milk
wasDerivedFrom
wasDerivedFrom
Butter
Cook
Icing
Used(ingredient)
wasControlledBy
(mixer)
wasControlledBy
Used
Flour
Used(ingredient)
Sugar
Used(ingredient)
Mix
wasTriggeredBy*
Decorate
wasGeneratedBy
(coating)
Eggs
Used(ingredient)
Used
(tool)
Cake
Hand
Mixer

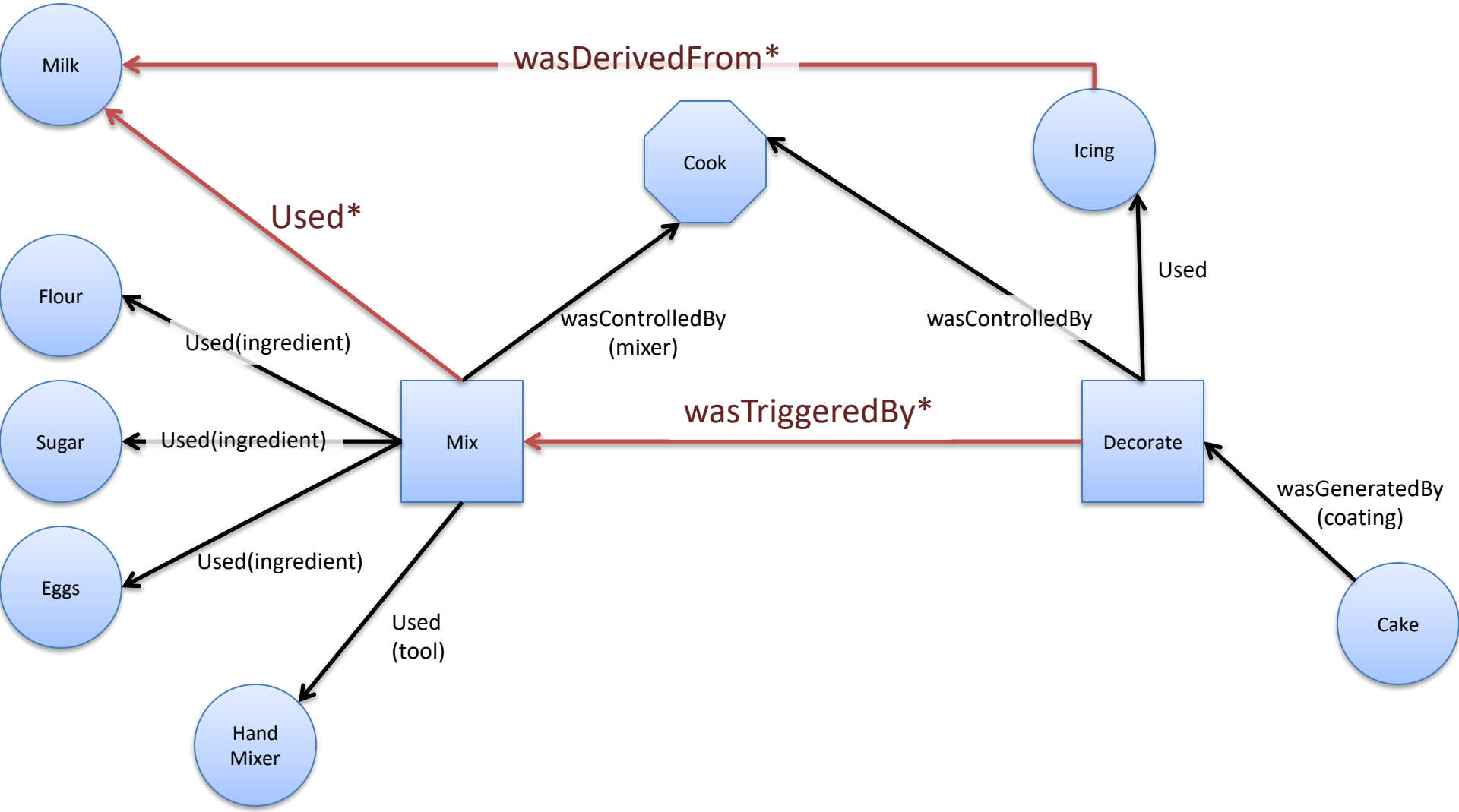
Inferência


Milk
wasDerivedFrom*
wasDerivedFrom
Butter
Cook
Icing
Flour
Used(ingredient)
wasControlledBy
(mixer)
wasControlledBy
Used
Used(ingredient)
Sugar
Used(ingredient)
Mix
wasTriggeredBy*
Decorate
wasGeneratedBy
(coating)
Eggs
Used(ingredient)
Used
(tool)
Cake
Hand
Mixer

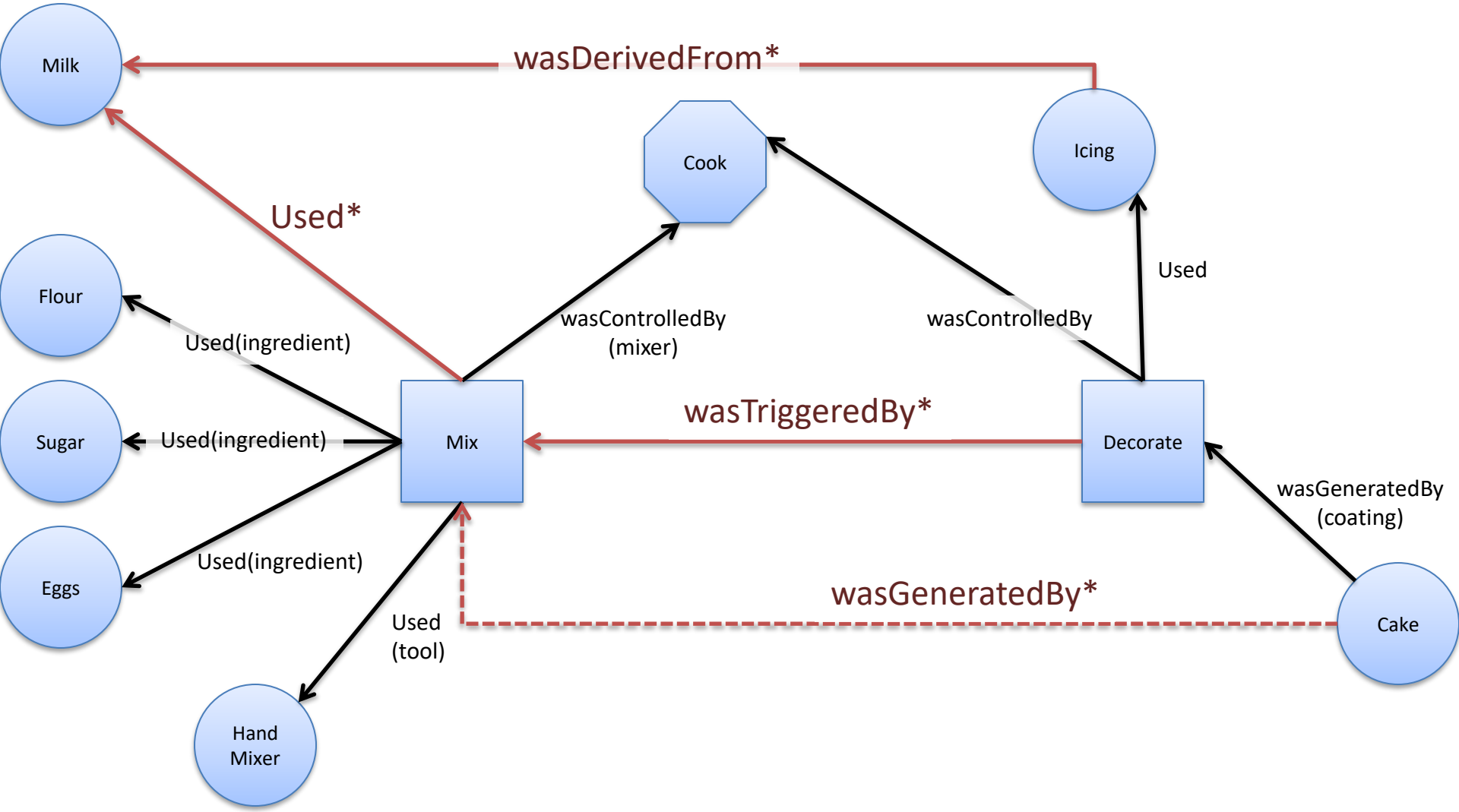
Inferência


Used*
Milk
wasDerivedFrom*
wasDerivedFrom
Butter
Cook
Icing
Used(ingredient)
Flour
wasControlledBy
(mixer)
wasControlledBy
Used
Used(ingredient)
Sugar
Used(ingredient)
Mix
wasTriggeredBy*
Decorate
wasGeneratedBy
(coating)
Used(ingredient)
Eggs
Used
(tool)
Cake
Hand
Mixer

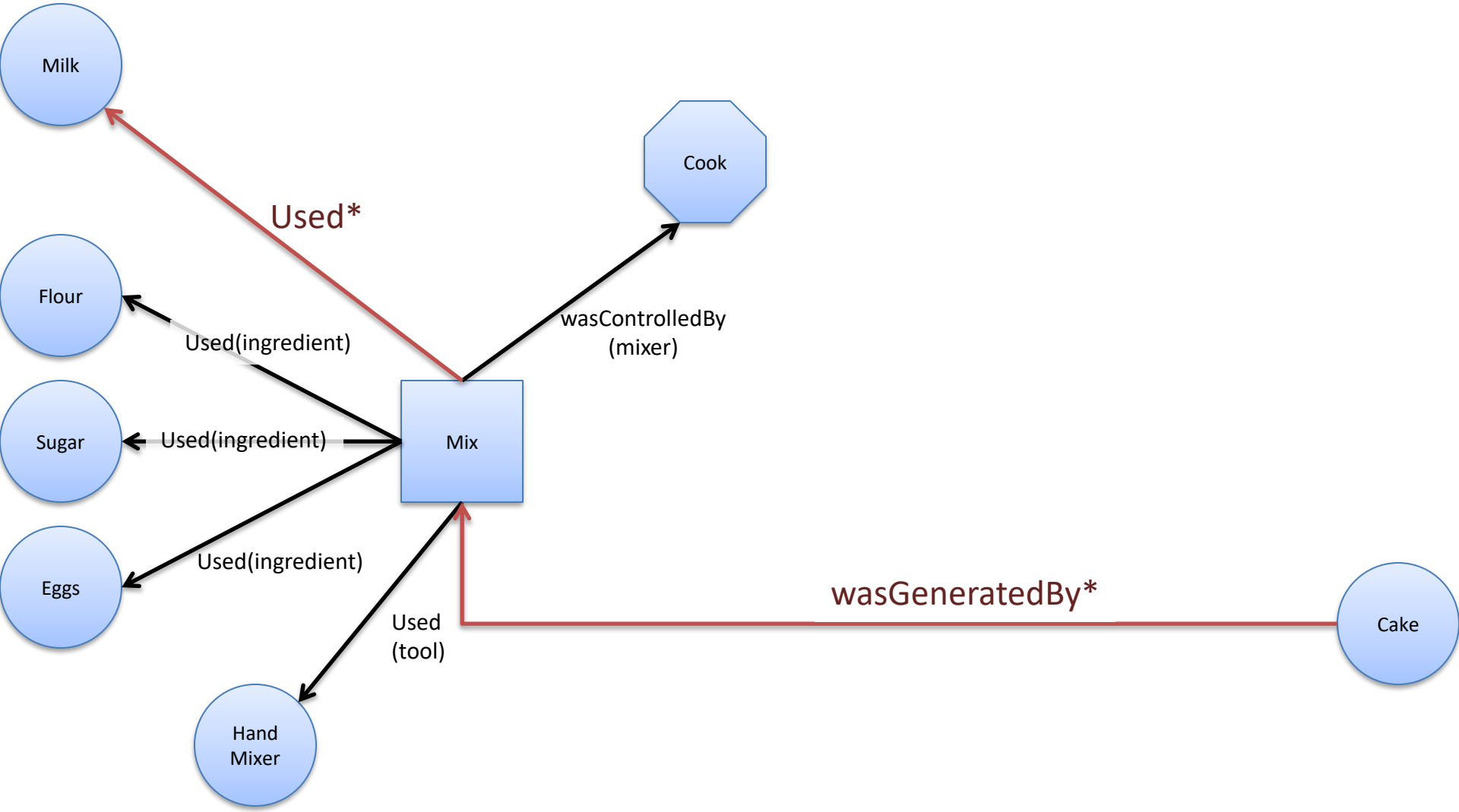
Inferência


Milk
wasDerivedFrom*
Cook
Icing
Used*
Flour
Used(ingredient)
wasControlledBy
(mixer)
wasControlledBy
Used
Sugar
Used(ingredient)
Mix
wasTriggeredBy*
Decorate
wasGeneratedBy
(coating)
Eggs
Used(ingredient)
Used
(tool)
Cake
Hand
Mixer

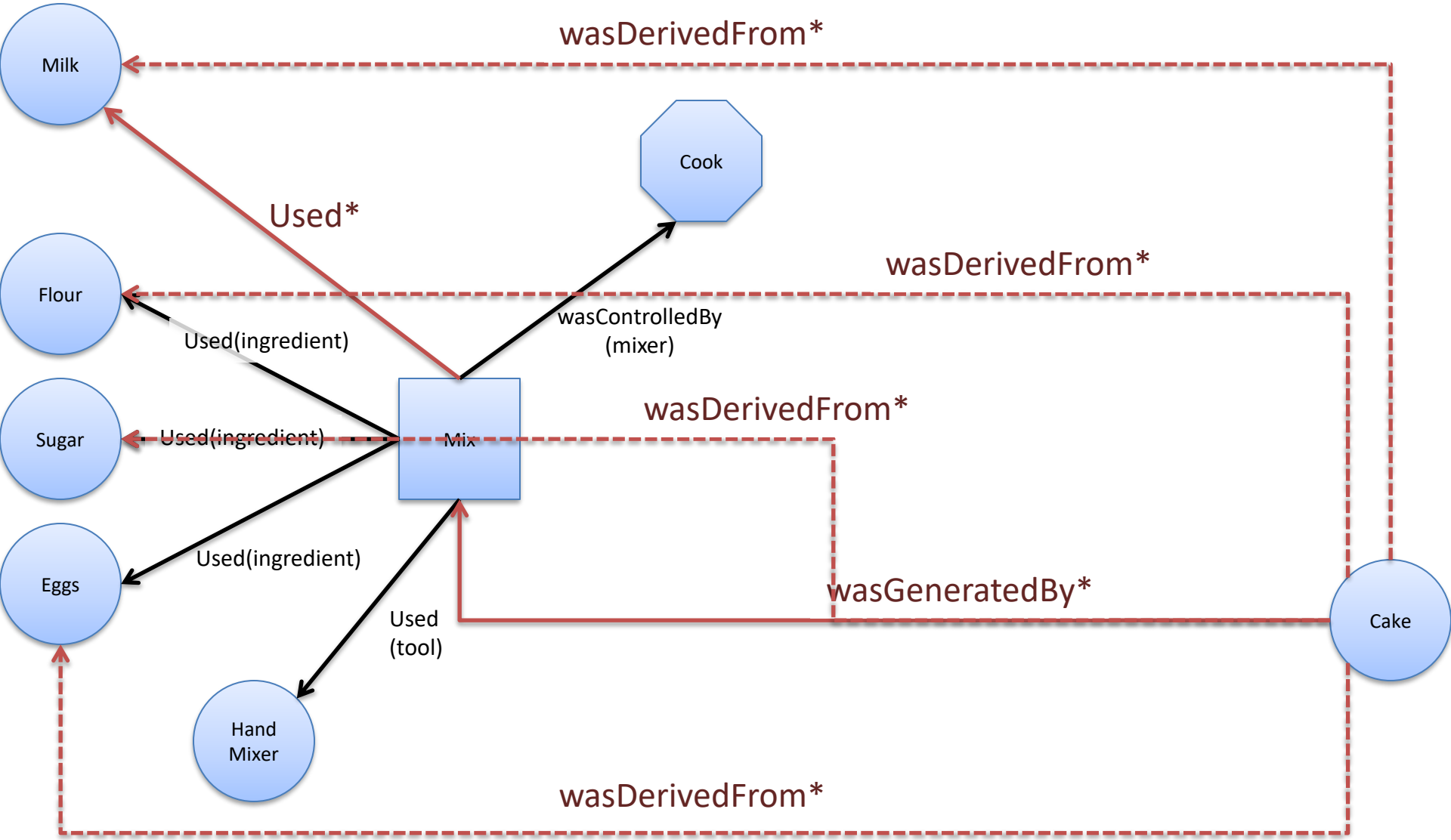
Inferência


Milk
wasDerivedFrom*
Cook
Icing
Used*
Flour
Used(ingredient)
wasControlledBy
(mixer)
wasControlledBy
Used
Sugar
Used(ingredient)
Mix
wasTriggeredBy*
Decorate
wasGeneratedBy
(coating)
Eggs
Used(ingredient)
wasGeneratedBy*
Used
(tool)
Cake
Hand
Mixer

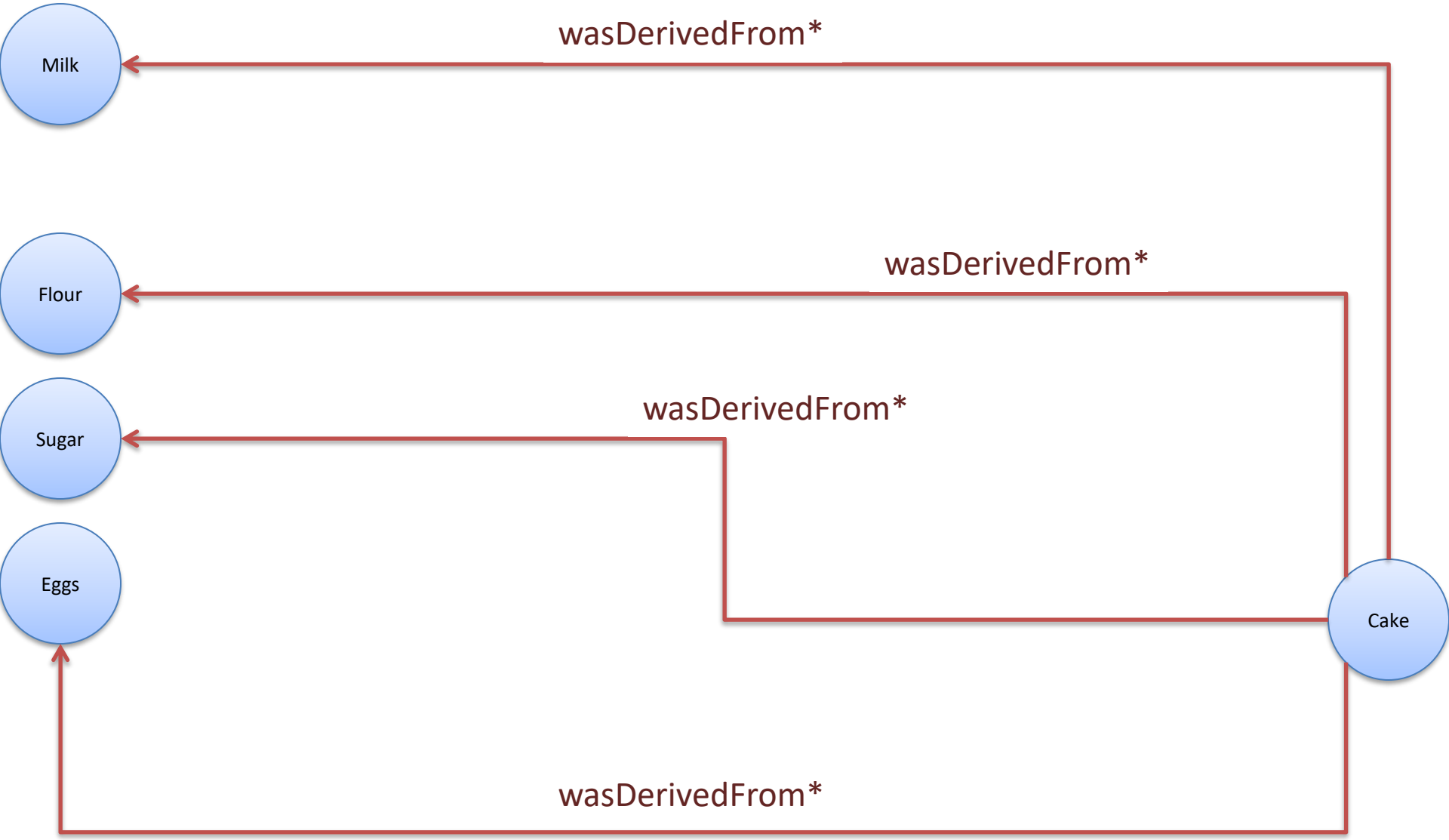
Inferência


Milk
Cook
Used*
Flour
Used(ingredient)
wasControlledBy
(mixer)
Sugar
Used(ingredient)
Mix
Eggs
Used(ingredient)
wasGeneratedBy*
Cake
Used
(tool)
Hand
Mixer

Inferência


wasDerivedFrom*
Milk
Cook
Used*
Flour
wasDerivedFrom*
wasControlledBy
(mixer)
Used(ingredient)
Sugar
Used(ingredient)
Mix
wasDerivedFrom*
Eggs
Used(ingredient)
wasGeneratedBy*
Cake
Used
(tool)
Hand
Mixer
wasDerivedFrom*

Inferência


Milk
wasDerivedFrom*
Flour
wasDerivedFrom*
Sugar
wasDerivedFrom*
Eggs
Cake
wasDerivedFrom*

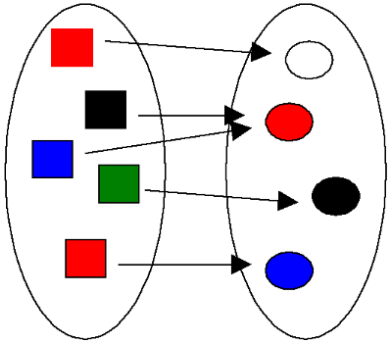
Instituto de Computação

Telemetria de Jogos
Proveniência
PinG
PinGU
Visualização
Considerações Finais

PING

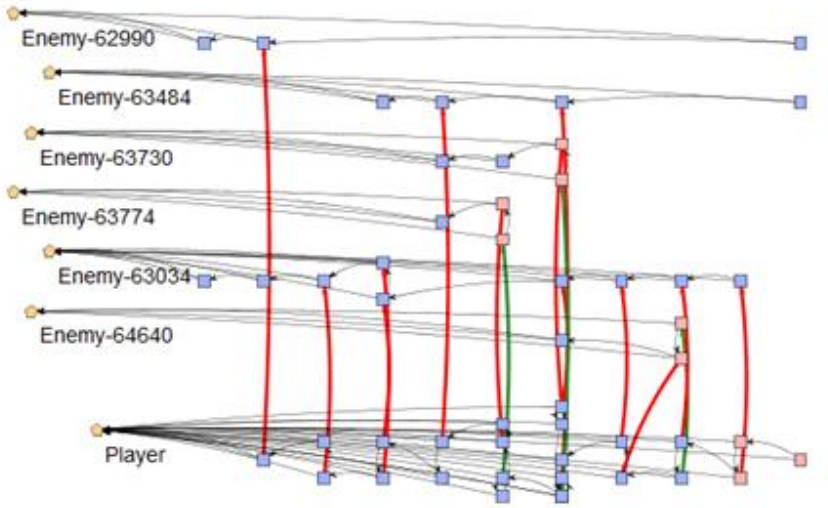
Provenance in Games (PinG)

Framework Conceitual



- Mapeamento de domínio
  - Proveniência para jogos
  - **Primeiro trabalho que trouxe proveniência**

- Coleta de dados
  - Proveniência
  - Relações causais



Proveniência em Jogos

- Entidades
  - Objetos
- Atividades
  - Ações
  - Eventos
- Agentes
  - NPCs
  - Jogador

Proveniência em Jogos

- Entidades
  - Objetos
- Atividades
  - Ações
  - Eventos
- Agentes
  - NPCs
  - Jogador

Proveniência em Jogos

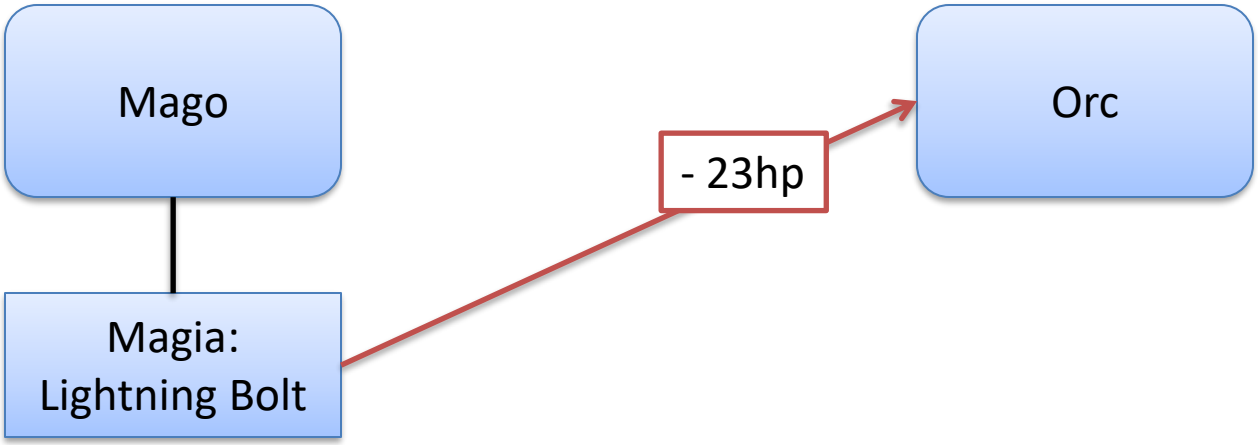
- Entidades
  - Objetos
- Atividades
  - Ações
  - Eventos
- Agentes
  - NPCs
  - Jogador



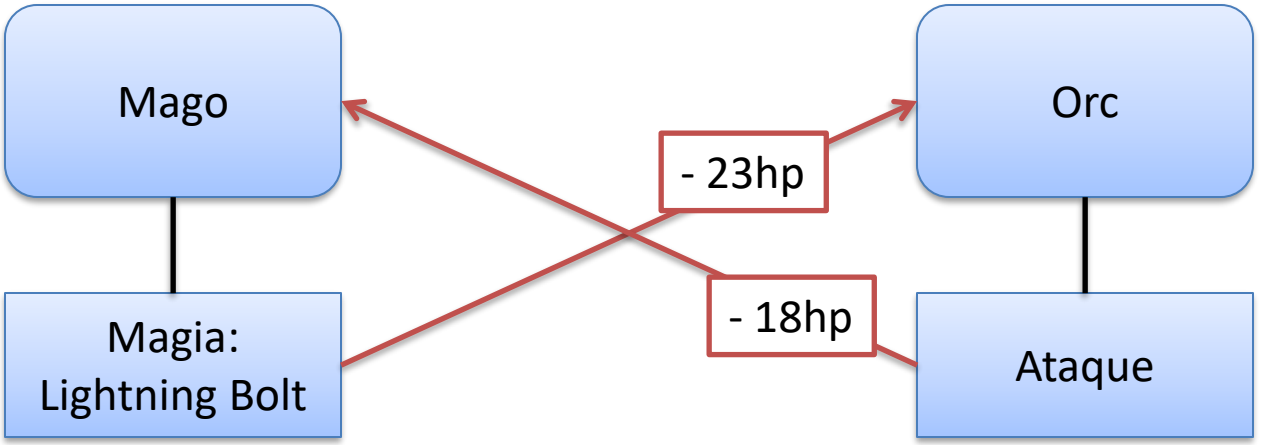



Proveniência em Jogos

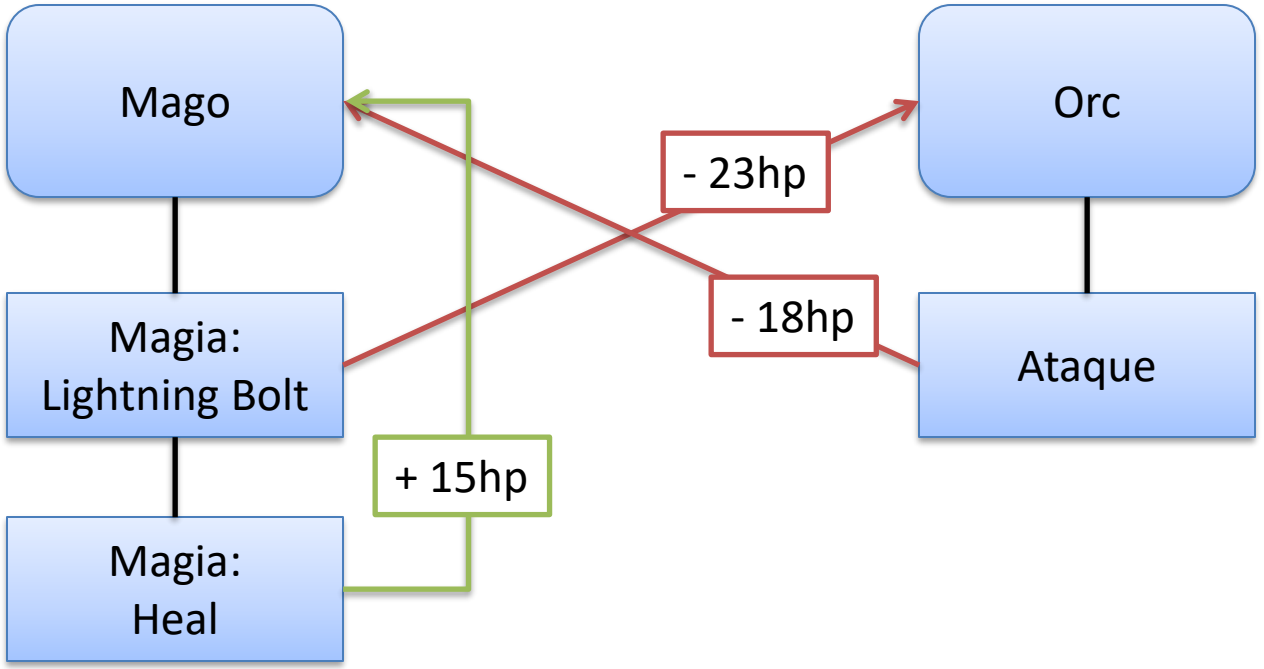
Proveniência em Jogos



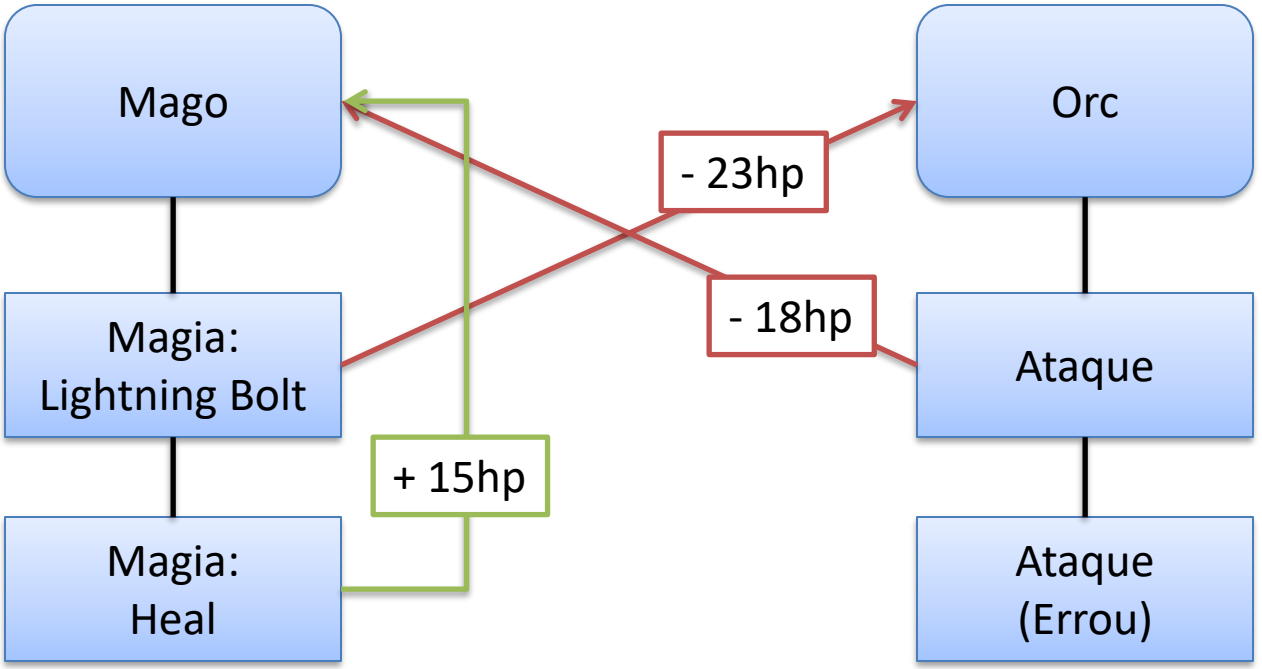
Proveniência em Jogos



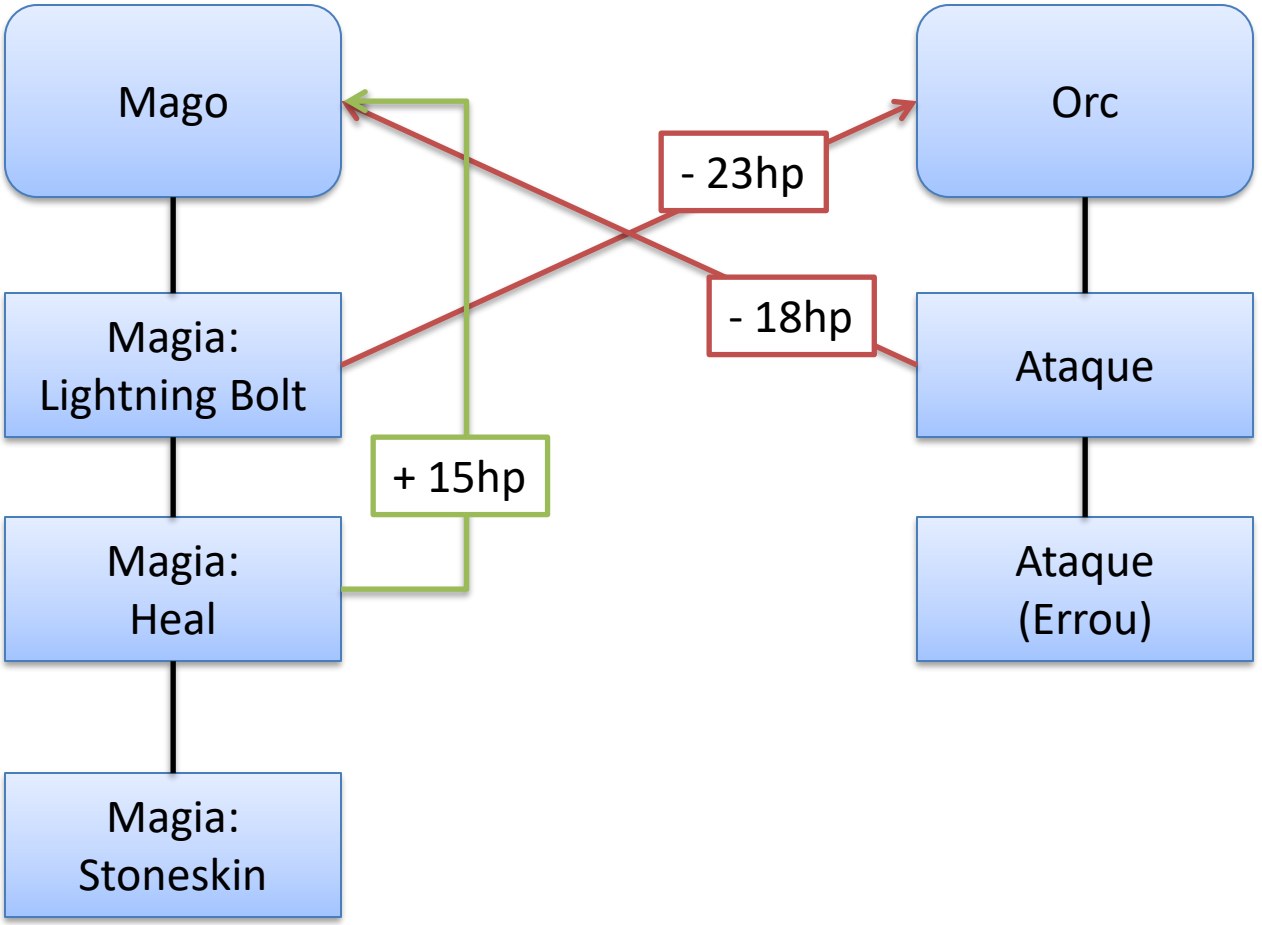
Proveniência em Jogos



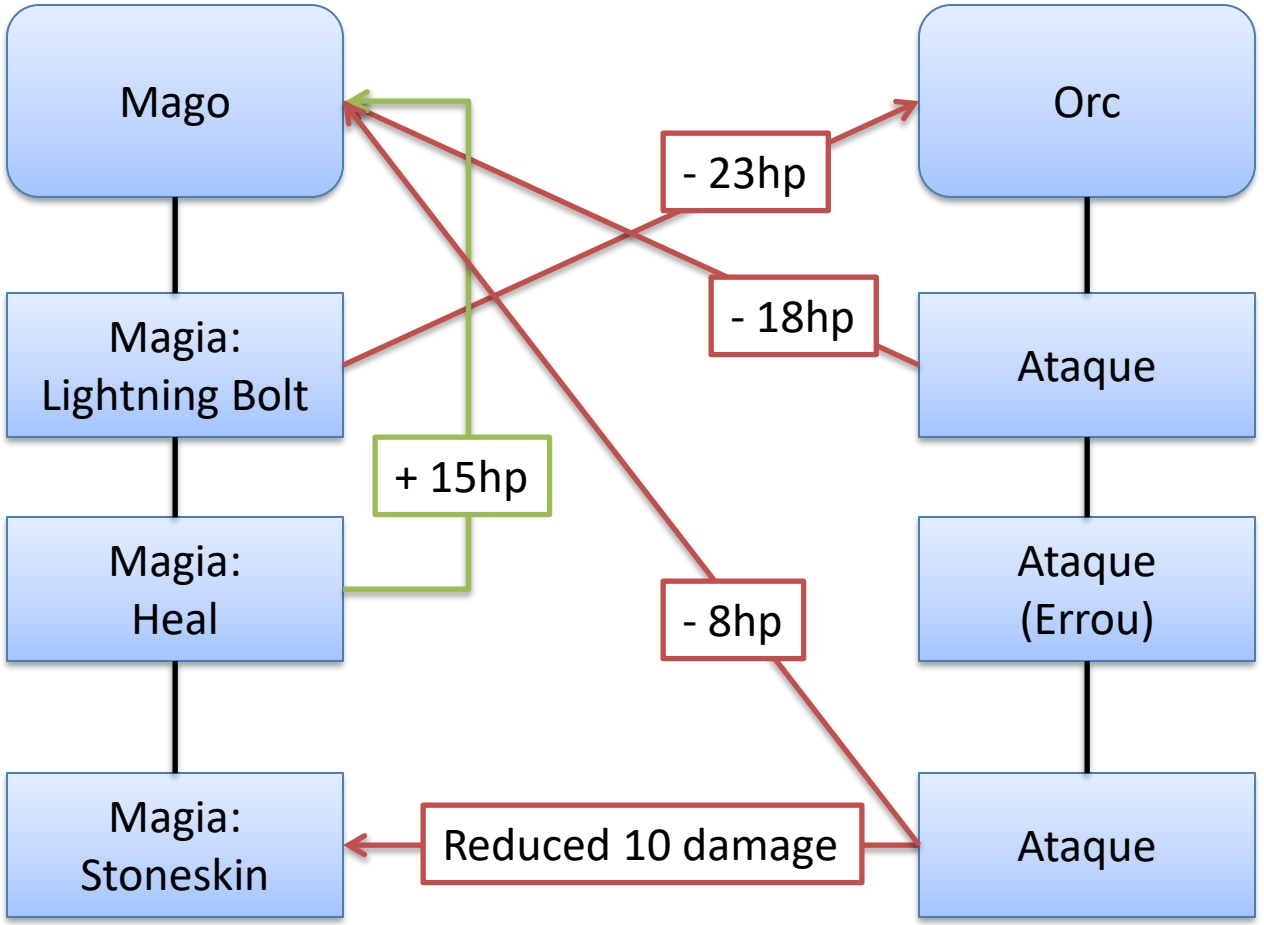
Proveniência em Jogos



Proveniência em Jogos



Proveniência em Jogos



Car Tutorial

https://www.assetstore.unity3d.com/en/#!/content/10

- Capturar dados de proveniência
- Gerar grafo para análise

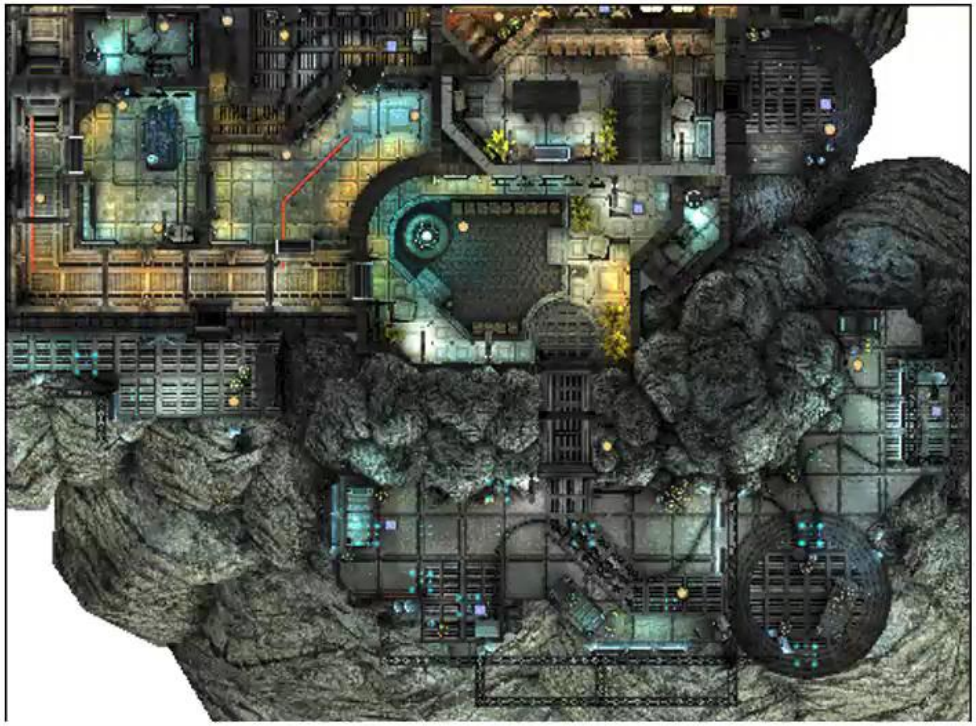
Grafo gerado no *Prov Viewer (Kohwalter et al., 2016)*
http://gems-uff.github.io/prov-viewer/

Angry Bots

https://www.assetstore.unity3d.com/en/#!/content/12175

- Capturar dados de proveniência
- Gerar grafo para análise

Grafo gerado no *Prov Viewer (Kohwalter et al., 2016)*
http://gems-uff.github.io/prov-viewer/



Mas como?

Instituto de Computação

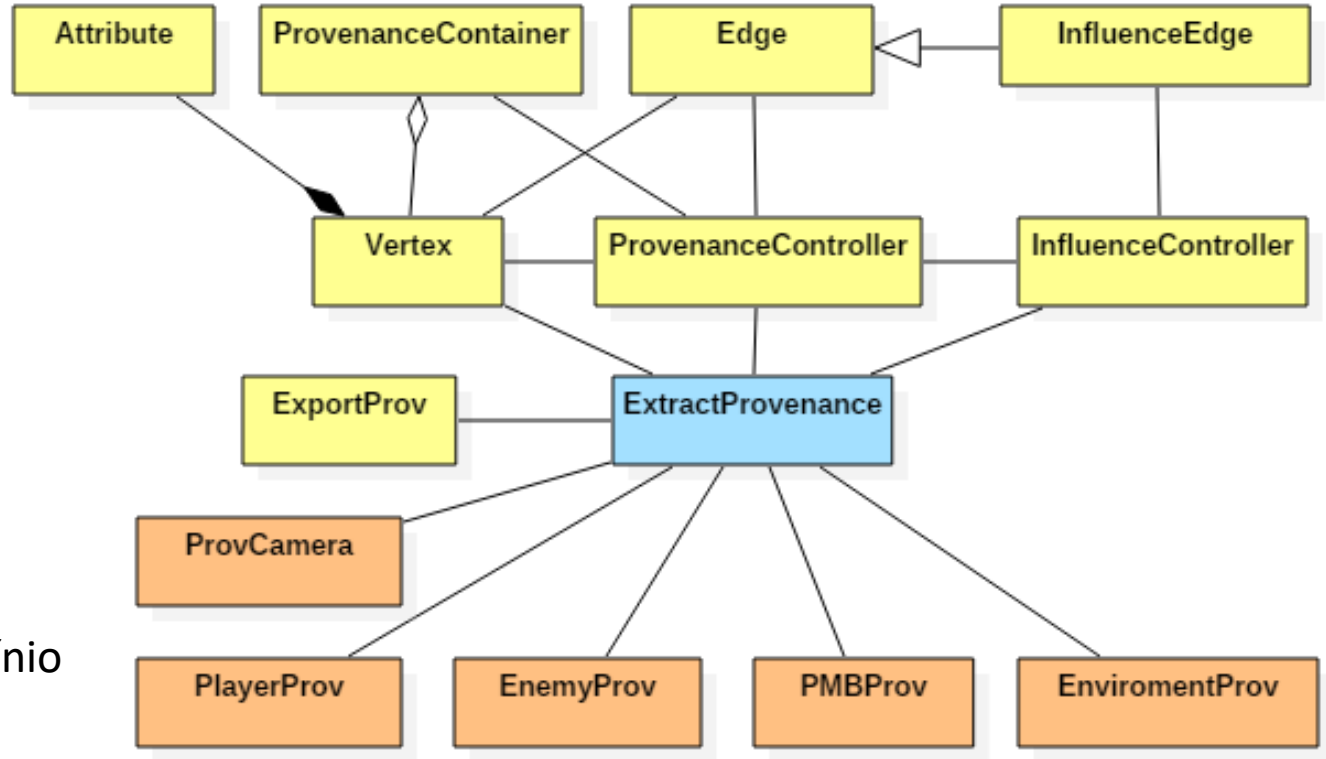
Telemetria de Jogos
Proveniência
PinG
PinGU
Visualização
Considerações Finais

PINGU

Provenance in Games for Unity PinGU

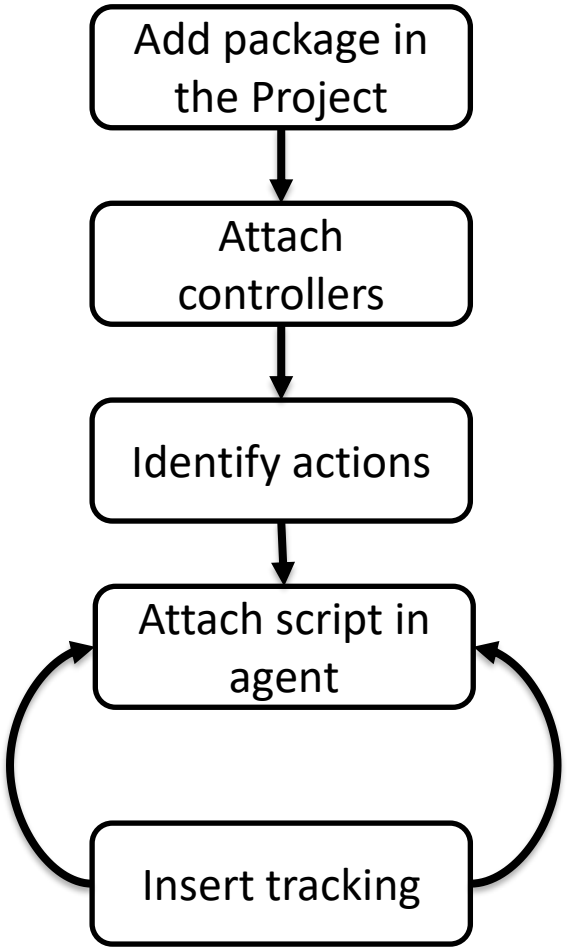
http://gems-uff.github.io/ping/

- Classes amarelas
  - Gerenciamento
- Classe azul
  - Coletor
- Classes laranjas
  - Específico de domínio



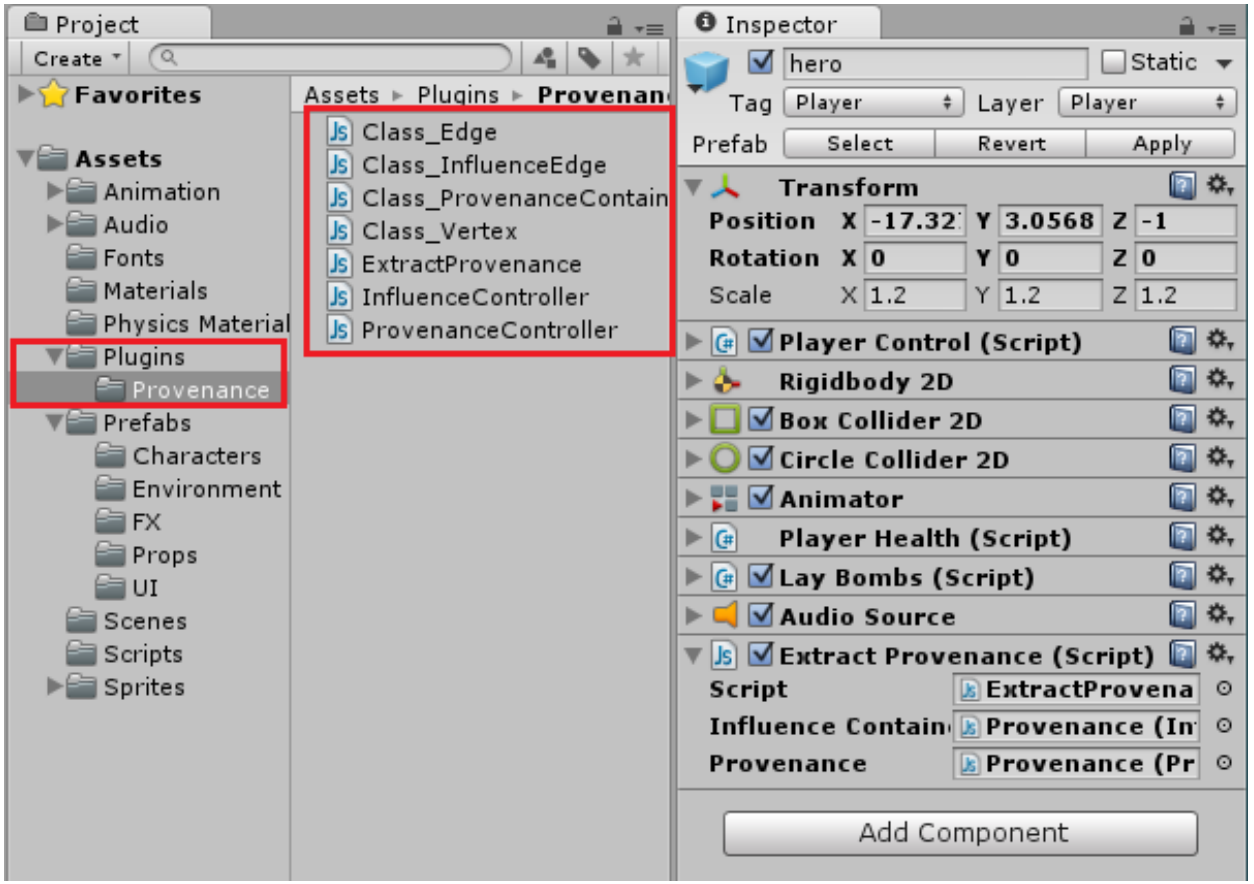
Integração

1. PinGU
2. Controladores de Proveniência
   - *Provenance Controller*
   - *Influence Controller*
3. Design do Jogo
   - Identificar ações
   - Identificar código
4. Extrator de proveniência
   - *Extract Provenance*
5. Funções de coleta
   - Especifico de domínio
   - Inserir no código



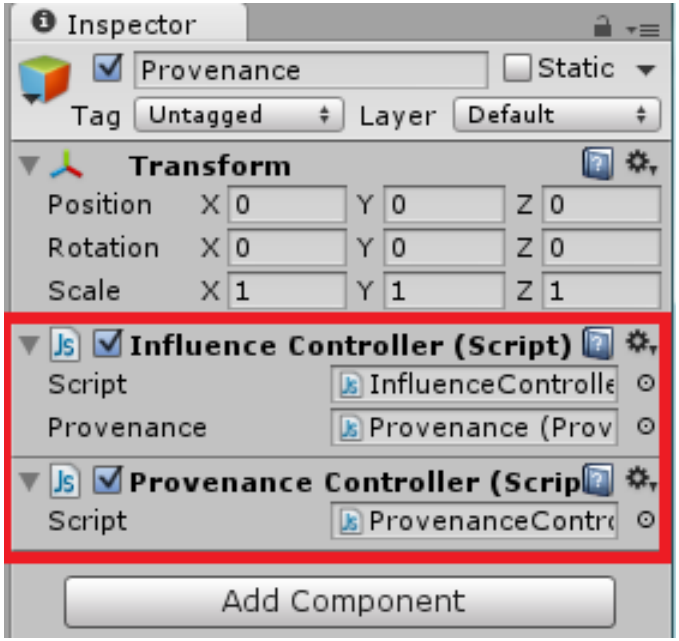
Integração

1. PinGU
2. Controladores de Proveniência
   - *Provenance Controller*
   - *Influence Controller*
3. Design do Jogo
   - Identificar ações
   - Identificar código
4. Extrator de proveniência
   - *Extract Provenance*
5. Funções de coleta
   - Especifico de domínio
   - Inserir no código



Integração

1. PinGU
2. Controladores de Proveniência
   - *Provenance Controller*
   - *Influence Controller*
3. Design do Jogo
   - Identificar ações
   - Identificar código
4. Extrator de proveniência
   - *Extract Provenance*
5. Funções de coleta
   - Especifico de domínio
   - Inserir no código



Integração

1. PinGU
2. Controladores de Proveniência
   - *Provenance Controller*
   - *Influence Controller*
3. Design do Jogo
   - Identificar ações
   - Identificar código
4. Extrator de proveniência
   - *Extract Provenance*
5. Funções de coleta
   - Especifico de domínio
   - Inserir no código

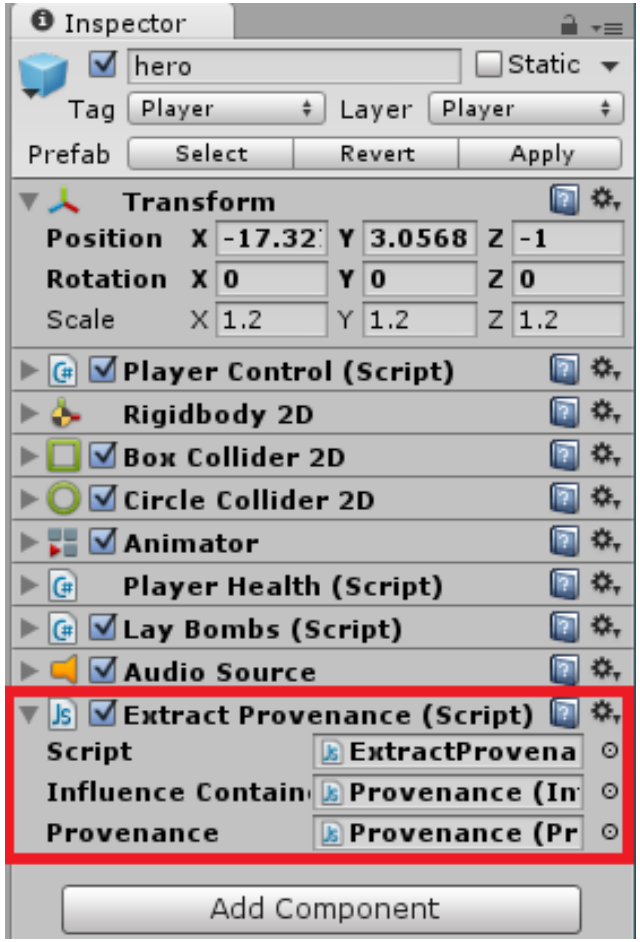
Exemplo

- **Enemy**
  - –Health (Attribute)
  - –Enemy Attack (Action)
    - •Player take Damage (Influence)
- **Player**
  - –Health (Attribute)
  - –Throw Bomb
    - •Secondary Attack (Action)
    - •Damage Enemy, Area of Effect (Influence)
  - –Fire Gun
    - •Primary Attack (Action)
    - •Spawn projectile
  - –Projectile
    - •Enemy Damage (Influence)
  - –Player Death (Action)

- **Item Spawner**
  - –Create Health Item (Action)
    - •Health Item (Object)
    - •Heal (Influence)
  - –Create Bomb Item (Action)
    - •Bomb container (Object)
    - •Increase Bomb ammunition (Influence)

Integração

1. PinGU
2. Controladores de Proveniência
   - *Provenance Controller*
   - *Influence Controller*
3. Design do Jogo
   - Identificar ações
   - Identificar código
4. Extrator de proveniência
   - *Extract Provenance*
5. Funções de coleta
   - Especifico de domínio
   - Inserir no código



Integração

1. PinGU
2. Controladores de Proveniência
   - *Provenance Controller*
   - *Influence Controller*
3. Design do Jogo
   - Identificar ações
   - Identificar código
4. Extrator de proveniência
   - *Extract Provenance*
5. Funções de coleta
   - Especifico de domínio
   - Inserir no código

Inserção da Coleta

```
function Fire () {
    if (weaponBehaviours[nextWeaponToFire]) {
        weaponBehaviours[nextWeaponToFire].SendMessage ("Fire");
        nextWeaponToFire = (nextWeaponToFire + 1) % weaponBehavi
        lastFireTime = Time.time;



    }
}
```

Integração

1. PinGU
2. Controladores de Proveniência
   - *Provenance Controller*
   - *Influence Controller*
3. Design do Jogo
   - Identificar ações
   - Identificar código
4. Extrator de proveniência
   - *Extract Provenance*
5. Funções de coleta
   - Especifico de domínio
   - Inserir no código

Inserção da Coleta

```
function Fire () {
    if (weaponBehaviours[nextWeaponToFire]) {
        weaponBehaviours[nextWeaponToFire].SendMessage ("Fire");
        nextWeaponToFire = (nextWeaponToFire + 1) % weaponBehavi
        lastFireTime = Time.time;

        // Provenance
        prov.Prov_Attack(damageAmount);
    }
}
```

Função de domínio

```
public string Prov_Attack(float damageAmount)
{
    Prov_GetEnemyAttributes();
    prov.NewActivityVertex("Attacking","");
    prov.GenerateInfluence("Player", this.GetInstanceID().ToString(),
    "Damage", (-damageAmount).ToString());
    return this.GetInstanceID().ToString();
}
```

Integração

1. PinGU
2. Controladores de Proveniência
   - *Provenance Controller*
   - *Influence Controller*
3. Design do Jogo
   - Identificar ações
   - Identificar código
4. Extrator de proveniência
   - *Extract Provenance*
5. Funções de coleta
   - Especifico de domínio
   - Inserir no código

Atributos de domínio

Inserção da Coleta

```
function Fire () {
    if (weaponBehaviours[nextWeaponToFire]) {
        weaponBehaviours[nextWeaponToFire].SendMessage ("Fire");
        nextWeaponToFire = (nextWeaponToFire + 1) % weaponBehavi
        lastFireTime = Time.time;

        // Provenance
        prov.Prov_Attack(damageAmount);
    }
}
```

Função de domínio

```
public string Prov_Attack(float damageAmount)
{
    Prov_GetEnemyAttributes();
    prov.NewActivityVertex("Attacking","");
    prov.GenerateInfluence("Player", this.GetInstanceID().ToString(),
    "Damage", (-damageAmount).ToString());
    return this.GetInstanceID().ToString();
}
```

Integração

1. PinGU
2. Controladores de Proveniência
   - *Provenance Controller*
   - *Influence Controller*
3. Design do Jogo
   - Identificar ações
   - Identificar código
4. Extrator de proveniência
   - *Extract Provenance*
5. Funções de coleta
   - Especifico de domínio
   - Inserir no código

Inserção da Coleta

```
function Fire () {
    if (weaponBehaviours[nextWeaponToFire]) {
        weaponBehaviours[nextWeaponToFire].SendMessage ("Fire");
        nextWeaponToFire = (nextWeaponToFire + 1) % weaponBehavi
        lastFireTime = Time.time;

        // Provenance
        prov.Prov_Attack(damageAmount);
    }
}
```

Função de domínio

Vértice

```
public string Prov_Attack(float damageAmount)
{
    Prov_GetEnemyAttributes();
    prov.NewActivityVertex("Attacking","");
    prov.GenerateInfluence("Player", this.GetInstanceID().ToString(),
    "Damage", (-damageAmount).ToString());
    return this.GetInstanceID().ToString();
}
```

Integração

1. PinGU
2. Controladores de Proveniência
   - *Provenance Controller*
   - *Influence Controller*
3. Design do Jogo
   - Identificar ações
   - Identificar código
4. Extrator de proveniência
   - *Extract Provenance*
5. Funções de coleta
   - Especifico de domínio
   - Inserir no código

Relação Causal

Inserção da Coleta

```
function Fire () {
    if (weaponBehaviours[nextWeaponToFire]) {
        weaponBehaviours[nextWeaponToFire].SendMessage ("Fire");
        nextWeaponToFire = (nextWeaponToFire + 1) % weaponBehavi
        lastFireTime = Time.time;

        // Provenance
        prov.Prov_Attack(damageAmount);
    }
}
```

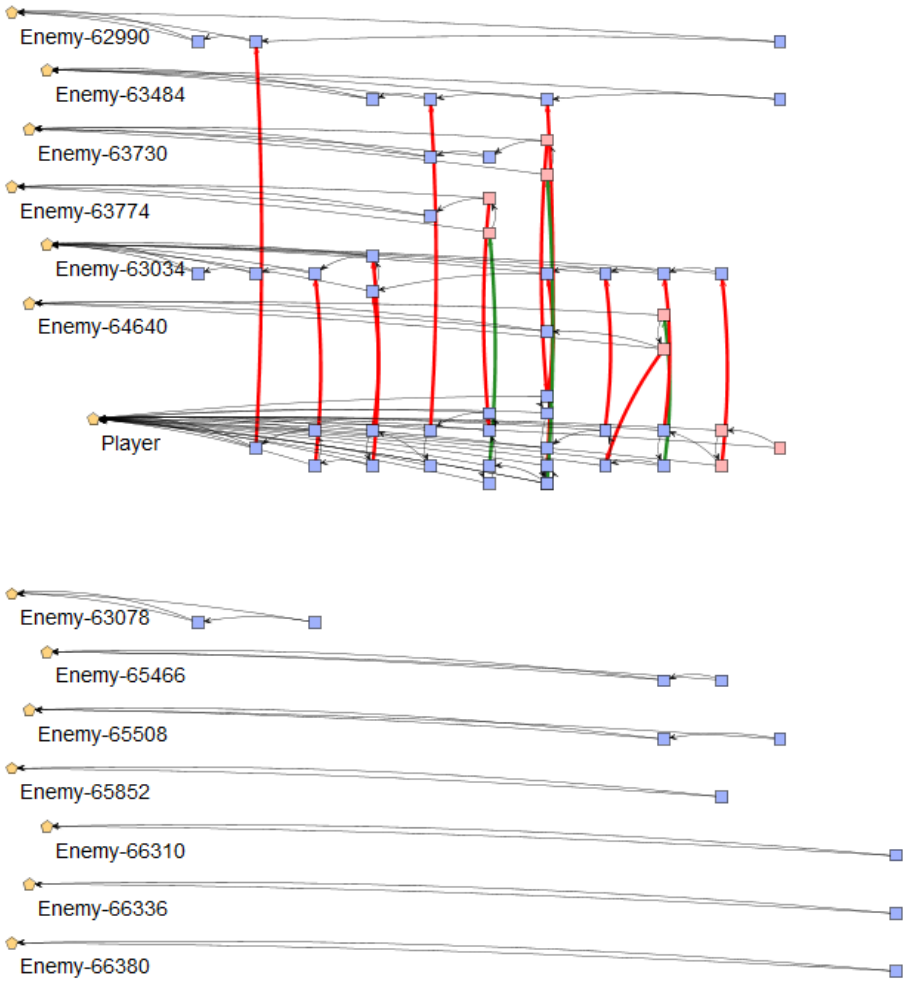
Função de domínio

```
public string Prov_Attack(float damageAmount)
{
    Prov_GetEnemyAttributes();
    prov.NewActivityVertex("Attacking","");
    prov.GenerateInfluence("Player", this.GetInstanceID().ToString(),
    "Damage", (-damageAmount).ToString());
    return this.GetInstanceID().ToString();
}
```

Grafo de Proveniência

- Cada agente
- Ações executadas
  - Quem executou
  - Relações causais

(arestas verticais)



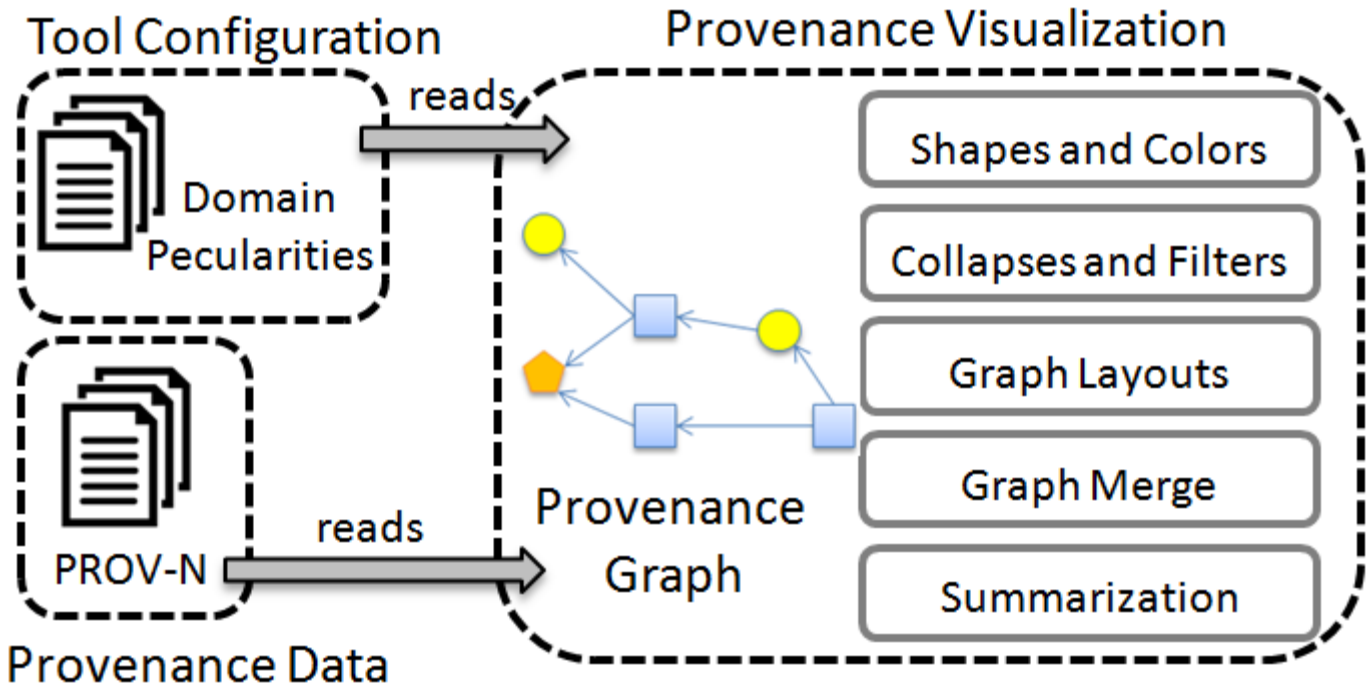
Telemetria de Jogos
Proveniência
PinG
PinGU
Visualização
Considerações Finais

VISUALIZAÇÃO

Prov Viewer

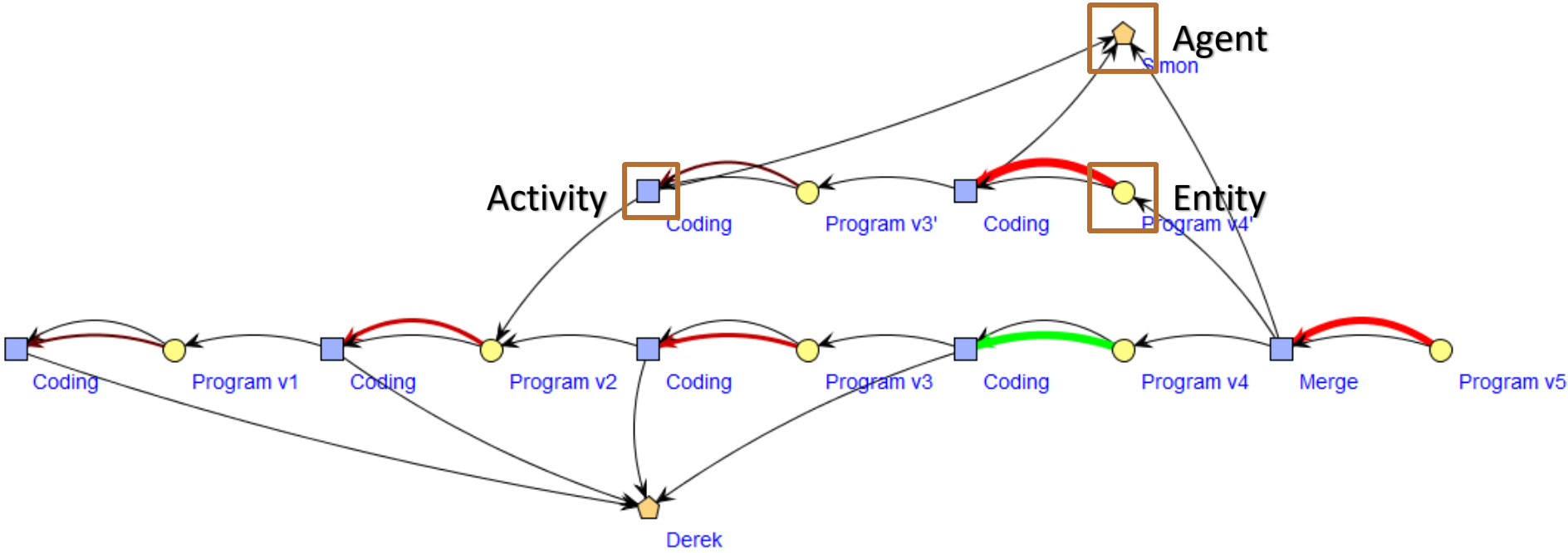
http://gems-uff.github.io/prov-viewer/

- Ferramenta de visualização de grafos

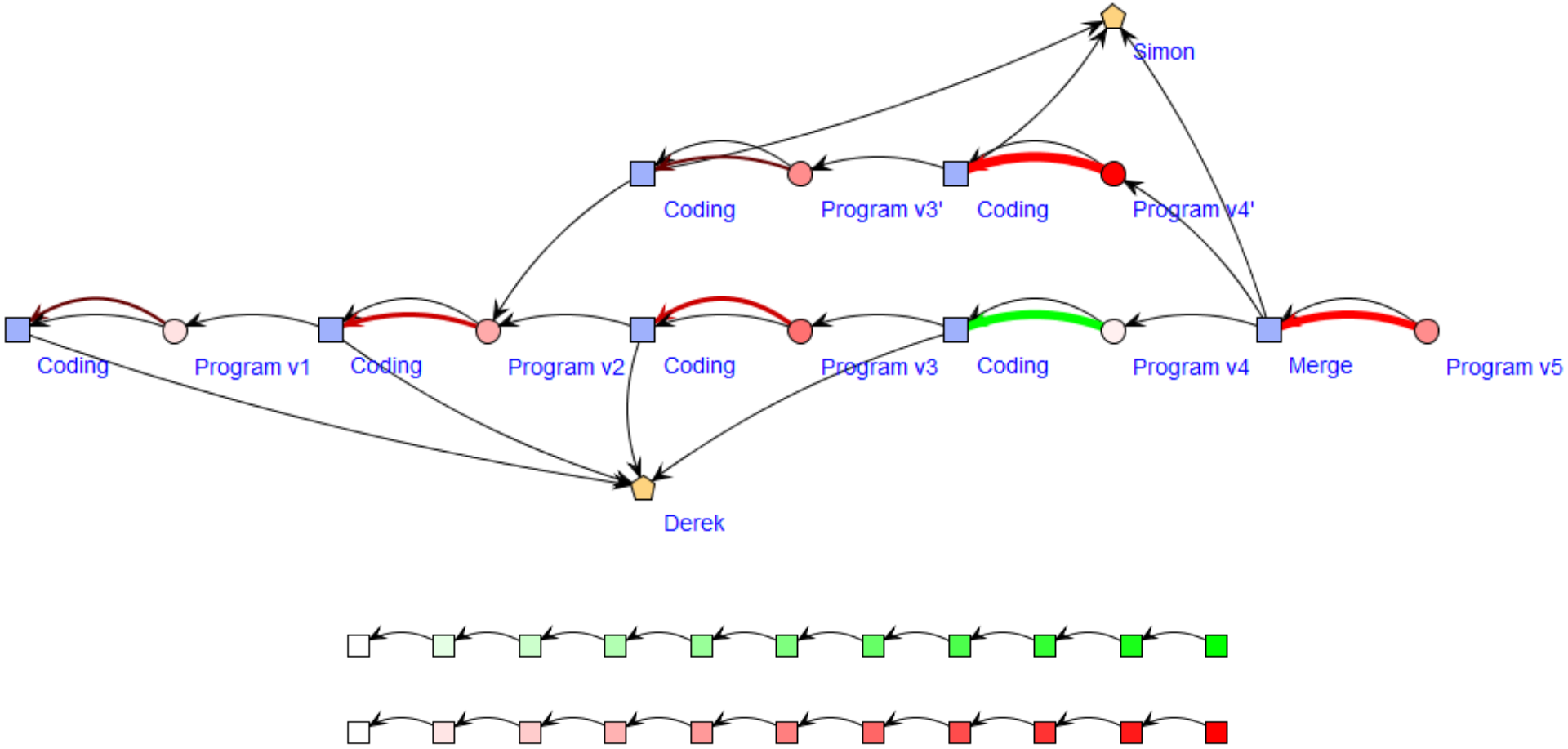


Formatos e Cores

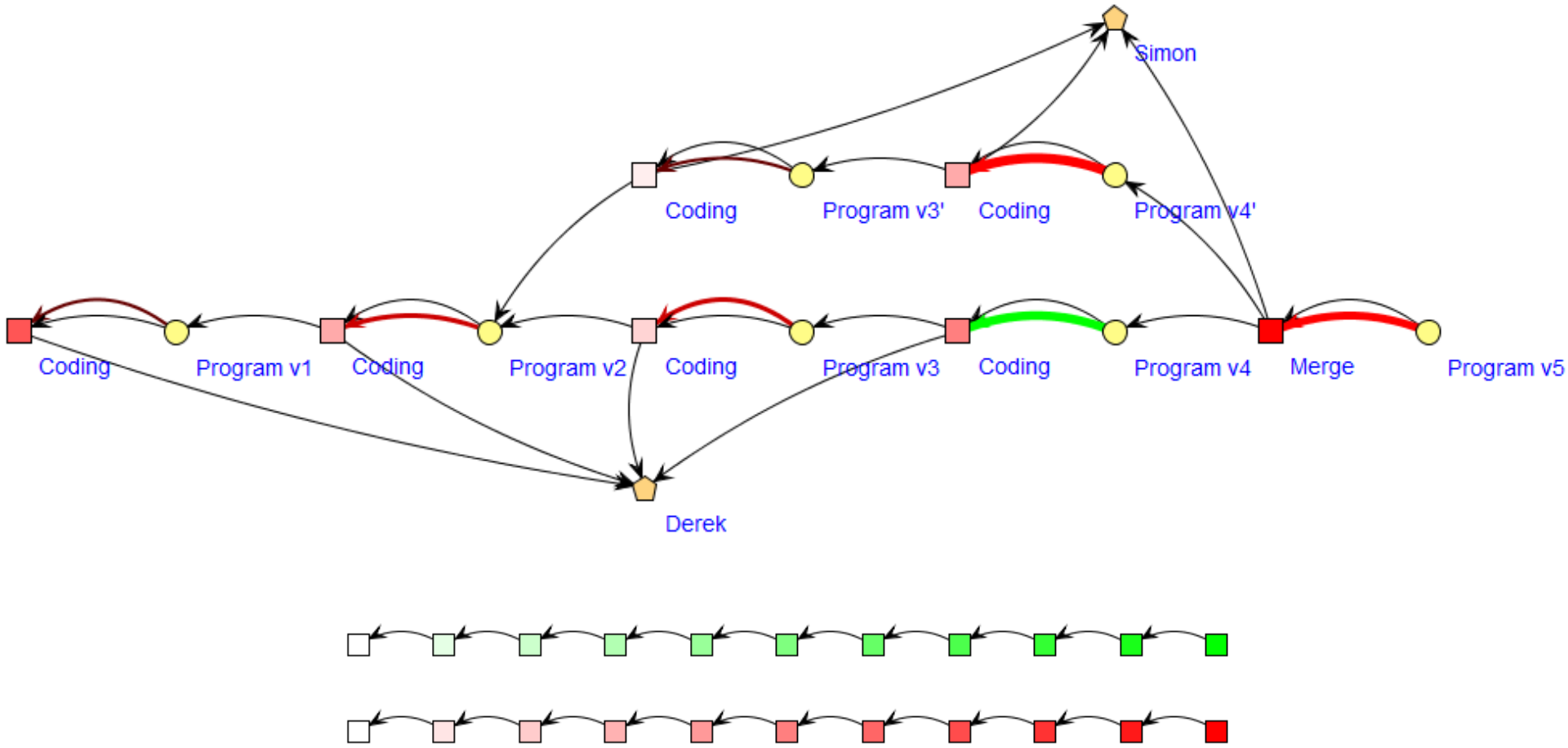
- PROV
  - Pentágono: Agente
  - Circulo: Entidade
  - Quadrado: Atividade
- Cores do PROV
- Color Schema
  - Traffic light baseado no metadado



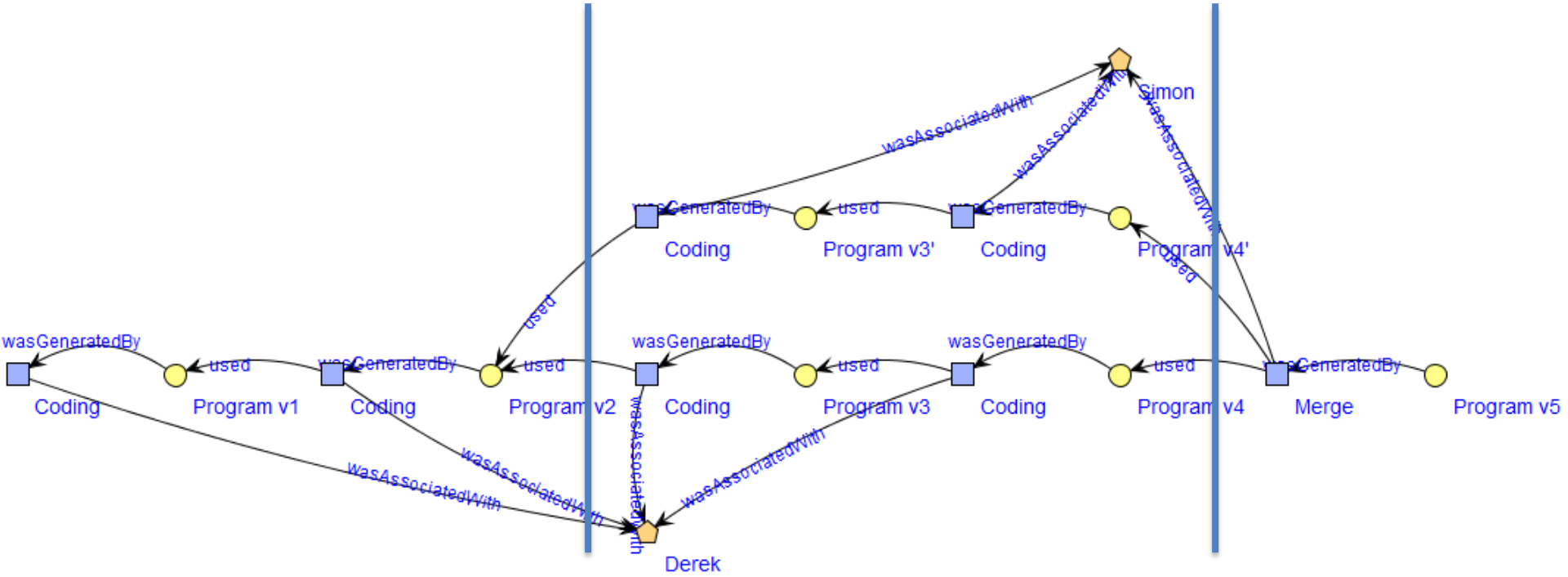
Formatos e Cores



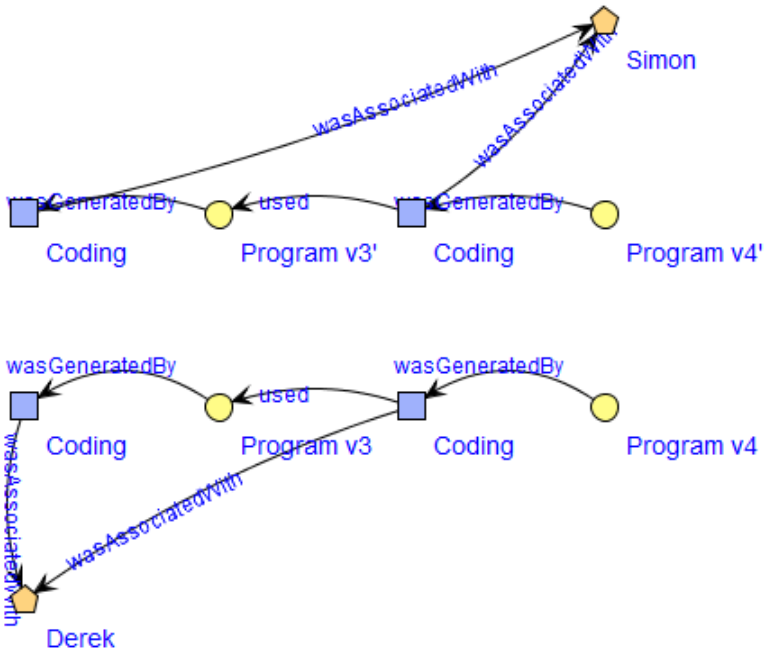
Formatos e Cores



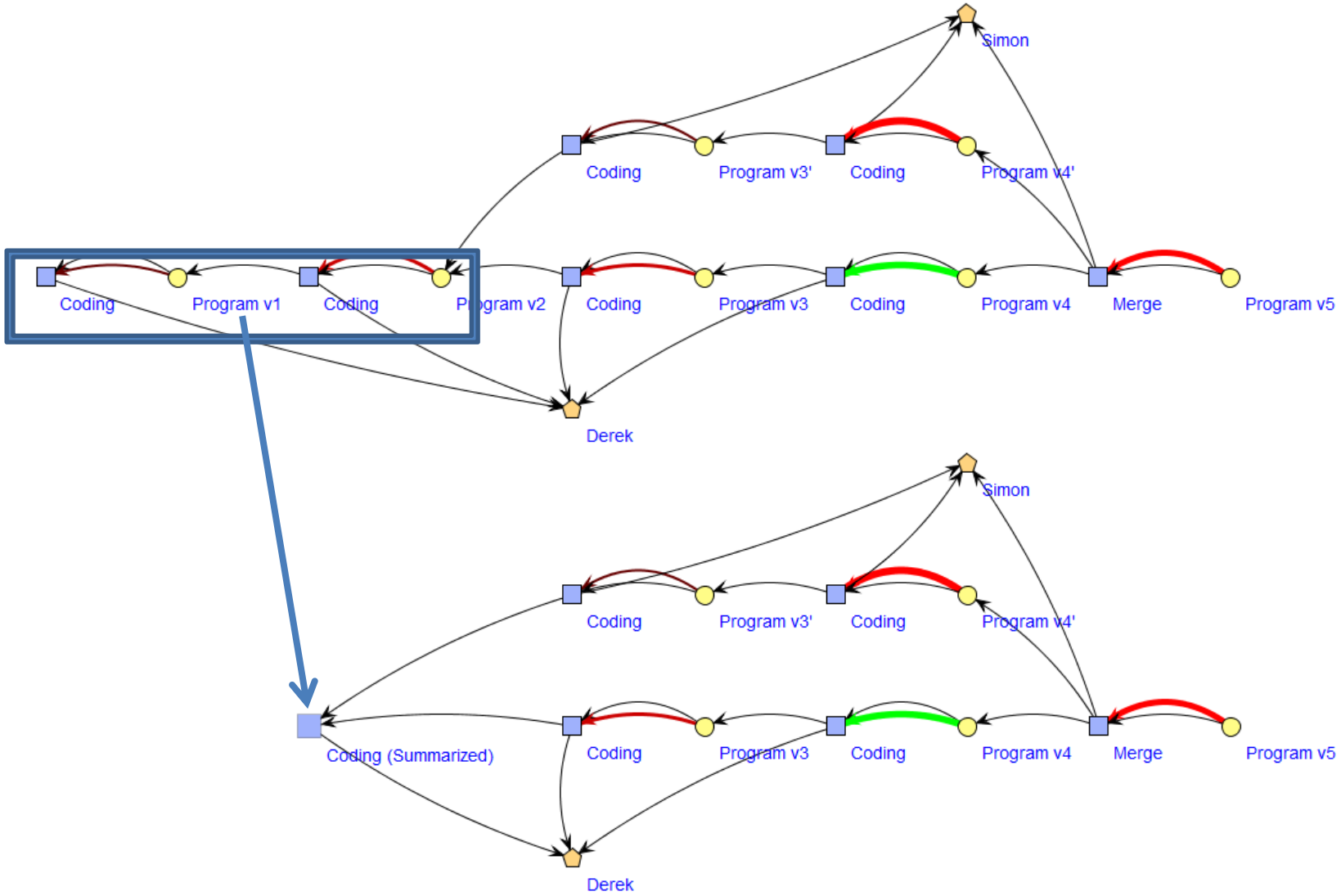
Filtro Temporal



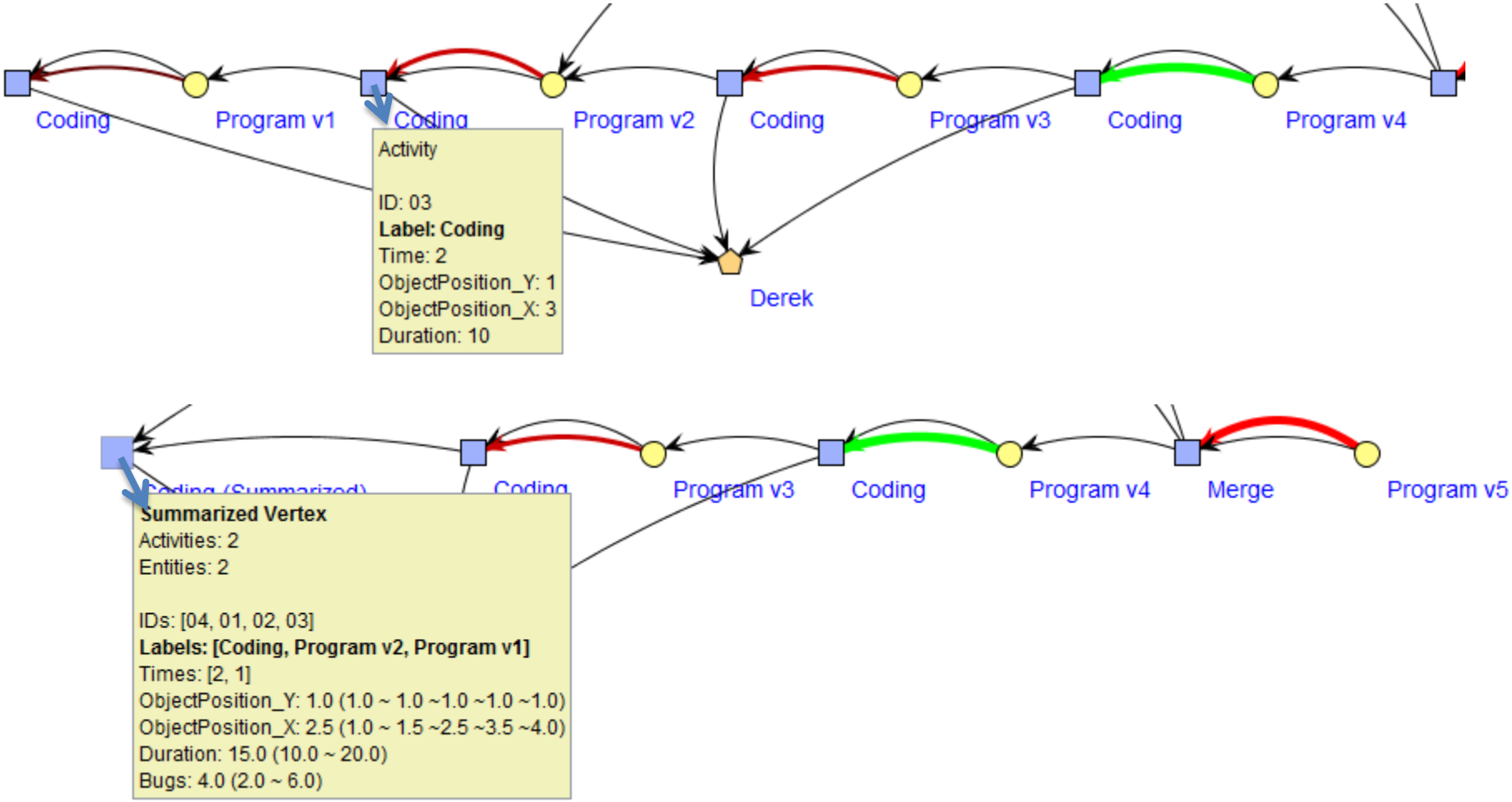
Filtro Temporal



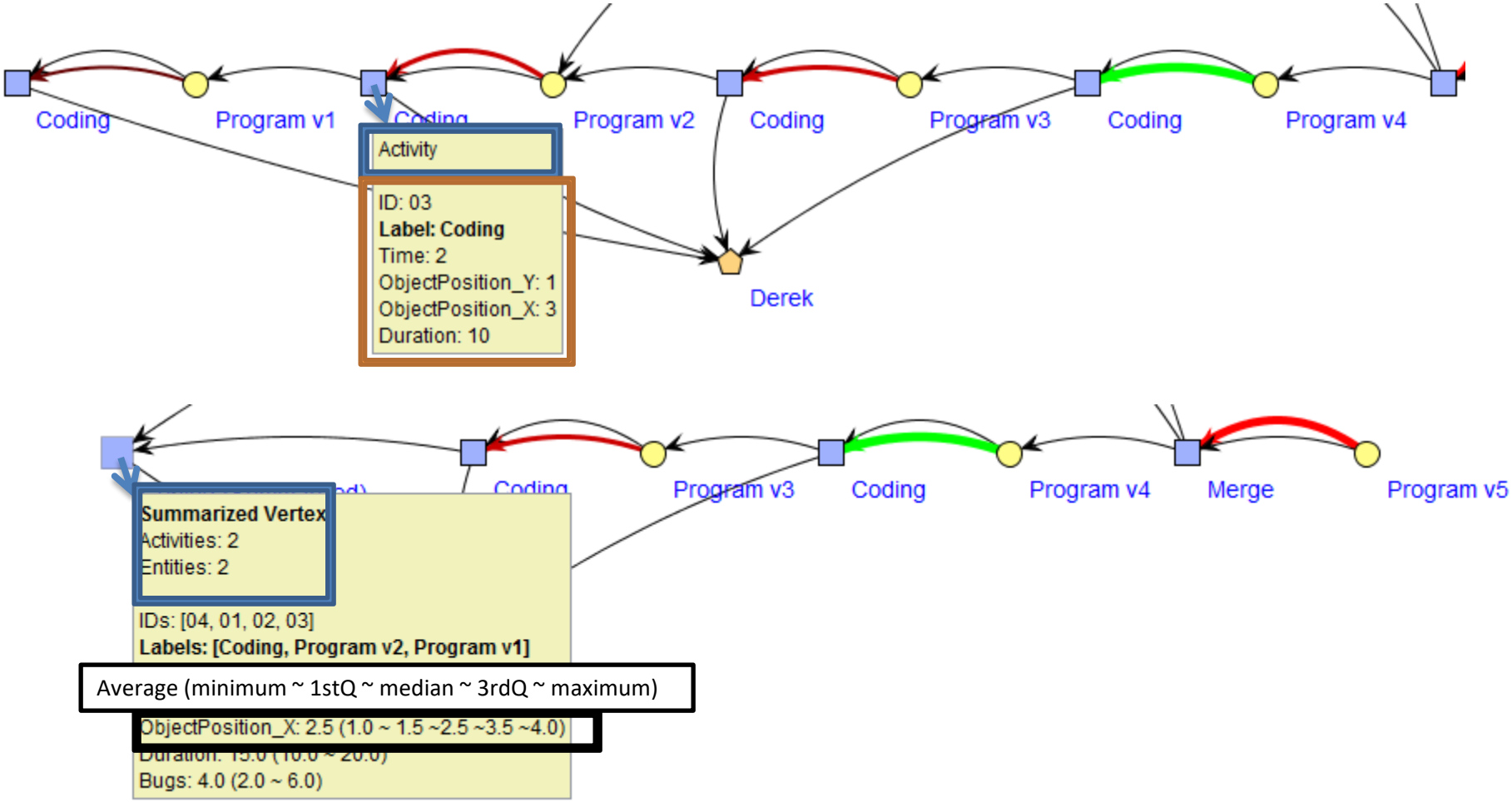
Sumarização



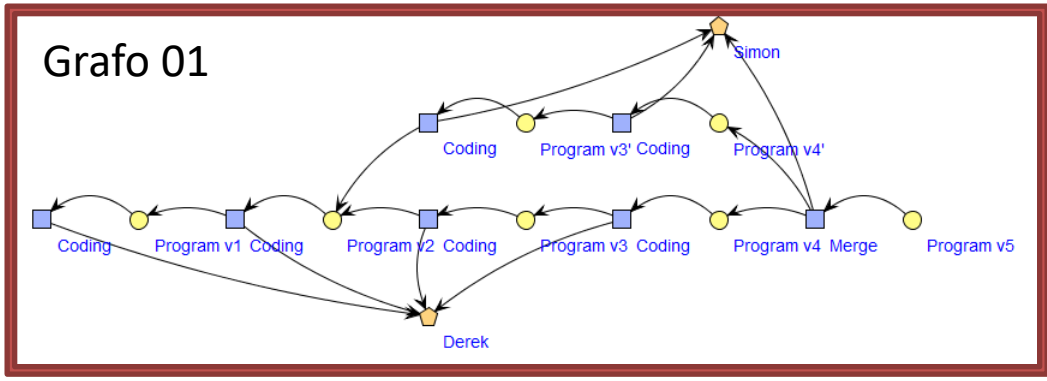
Metadado do vértice



Metadado do vértice



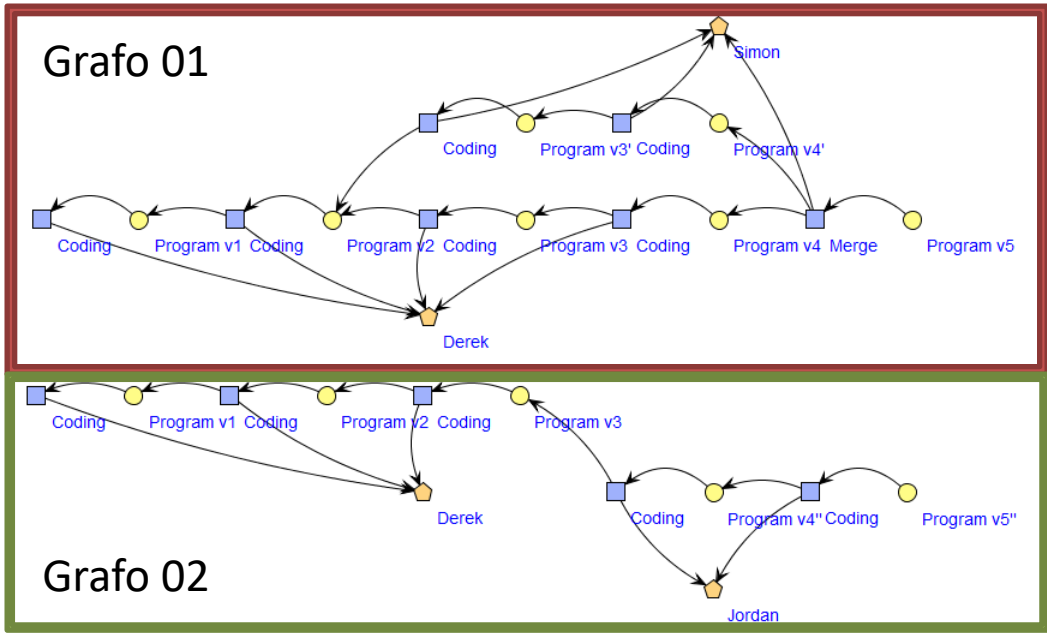
Sumarização Automática



- Combinar vértices semelhantes
  - Comparar vértices
  - Detectar similaridade
  - Sem perdas de informação
- Similaridade
  - Tipo do vértice
  - Atributos
  - Valores
- Entre grafos
  - Merge
  - Analisar várias sessões ou trials
- No mesmo grafo
  - Vértices Sequenciais
  - Deduplicação

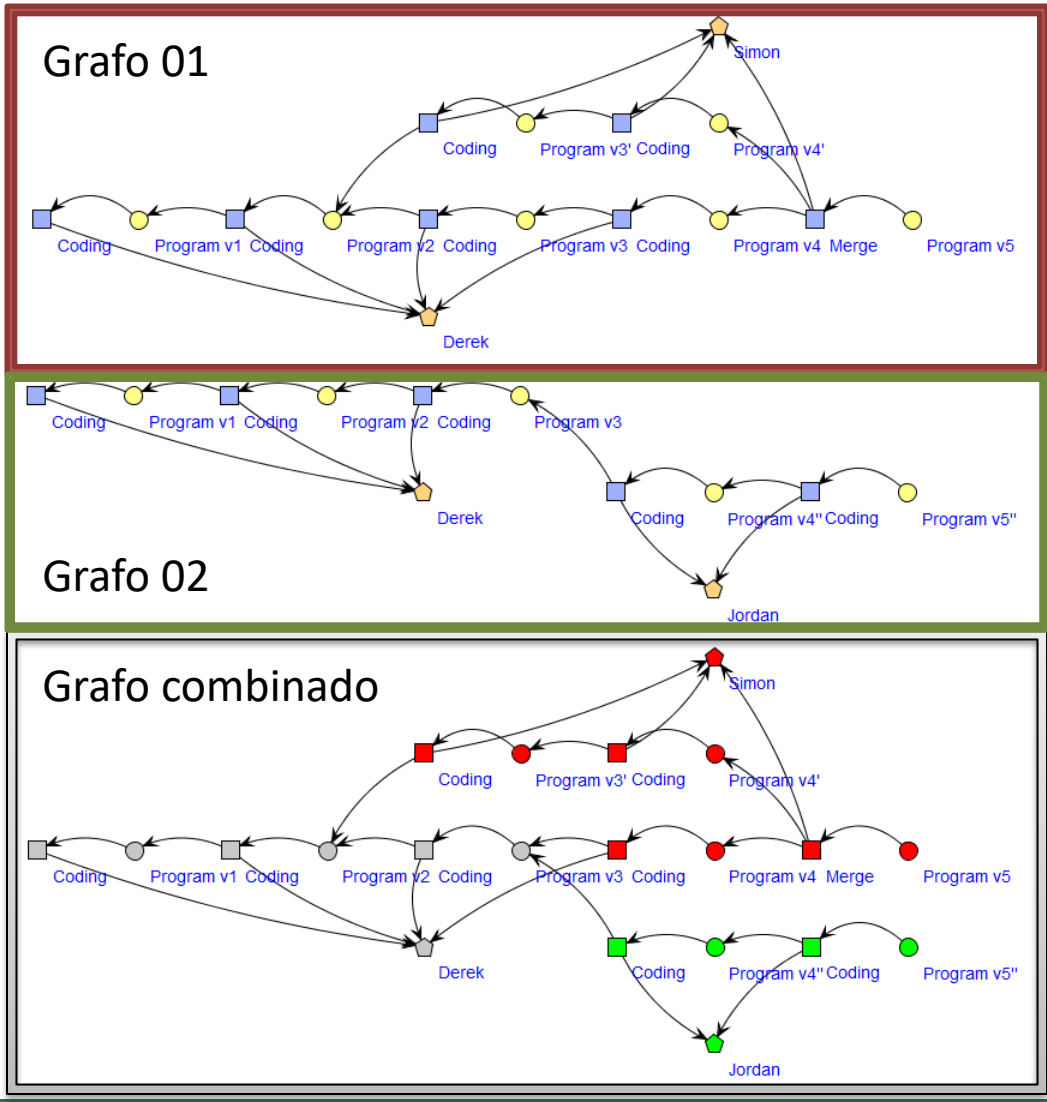
Sumarização Automática

- Combinar vértices semelhantes
  - Comparar vértices
  - Detectar similaridade
  - Sem perdas de informação
- Similaridade
  - Tipo do vértice
  - Atributos
  - Valores
- Entre grafos
  - Merge
  - Analisar várias sessões ou trials
- No mesmo grafo
  - Vértices Sequenciais
  - Deduplicação



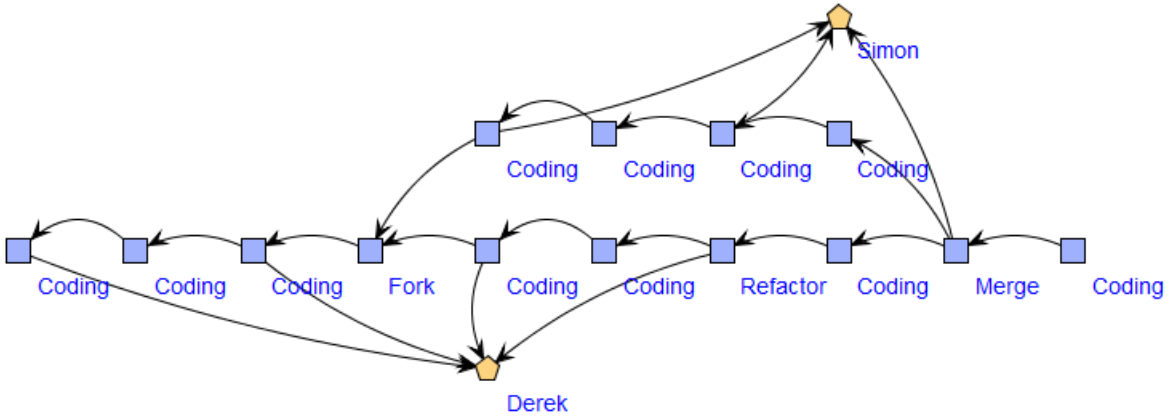
Sumarização Automática

- Combinar vértices semelhantes
  - Comparar vértices
  - Detectar similaridade
  - Sem perdas de informação
- Similaridade
  - Tipo do vértice
  - Atributos
  - Valores
- Entre grafos
  - Merge
  - Analisar várias sessões ou trials
- No mesmo grafo
  - Vértices Sequenciais
  - Deduplicação



Sumarização Automática

- Combinar vértices semelhantes
  - Comparar vértices
  - Detectar similaridade
  - Sem perdas de informação
- Similaridade
  - Tipo do vértice
  - Atributos
  - Valores

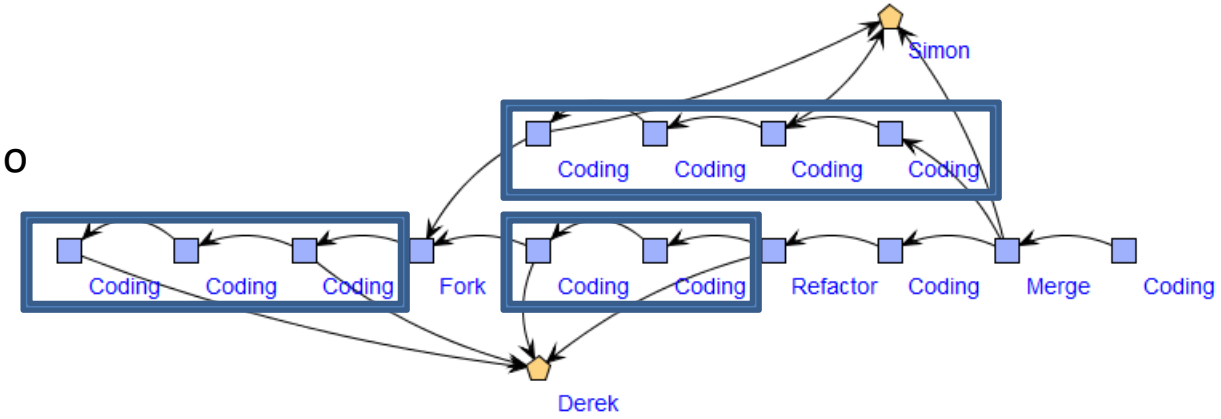


- Entre grafos
  - Merge
  - Analisar várias sessões ou trials

- No mesmo grafo
  - Vértices Sequenciais
  - Deduplicação

Sumarização Automática

- Combinar vértices semelhantes
  - Comparar vértices
  - Detectar similaridade
  - Sem perdas de informação
- Similaridade
  - Tipo do vértice
  - Atributos
  - Valores

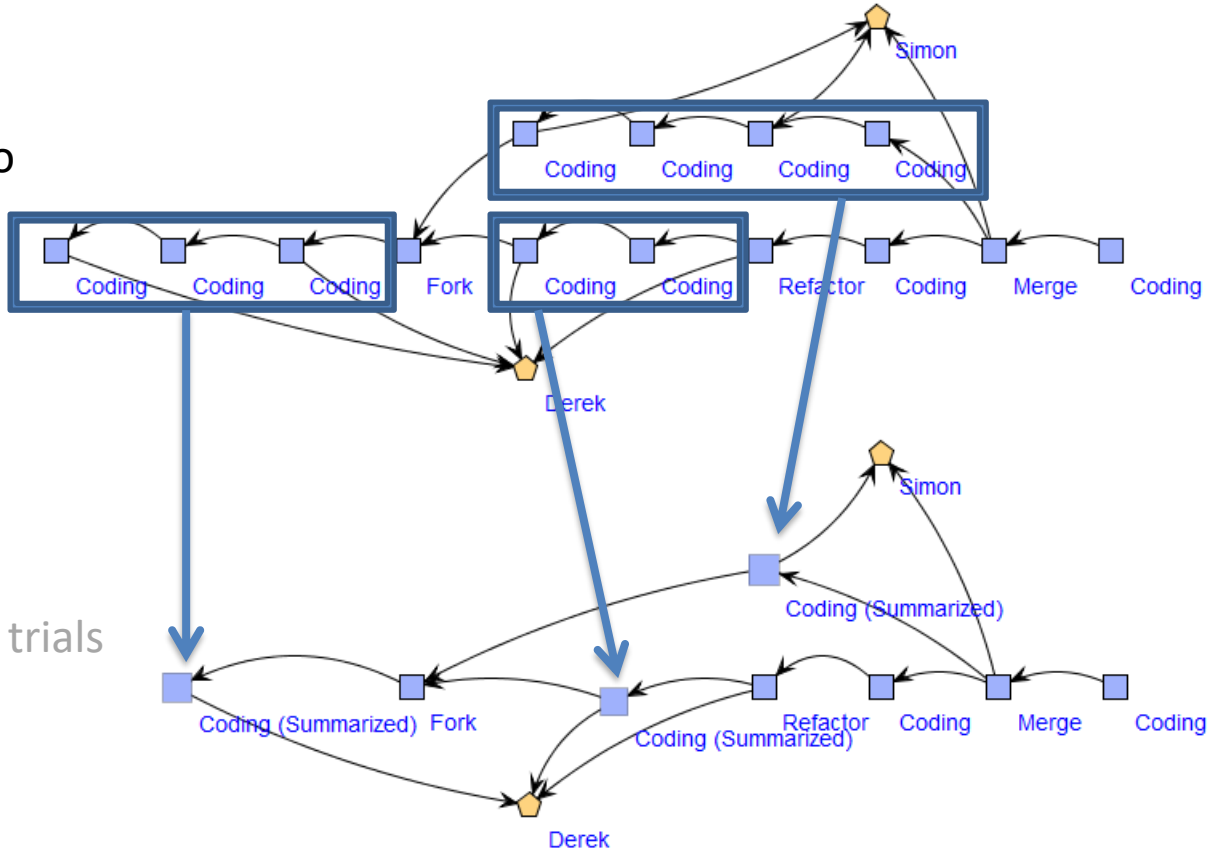


- Entre grafos
  - Merge
  - Analisar várias sessões ou trials

- No mesmo grafo
  - Vértices Sequenciais
  - Deduplicação

Sumarização Automática

- Combinar vértices semelhantes
  - Comparar vértices
  - Detectar similaridade
  - Sem perdas de informação
- Similaridade
  - Tipo do vértice
  - Atributos
  - Valores
- Entre grafos
  - Merge
  - Analisar várias sessões ou trials
- No mesmo grafo
  - Vértices Sequenciais
  - Deduplicação

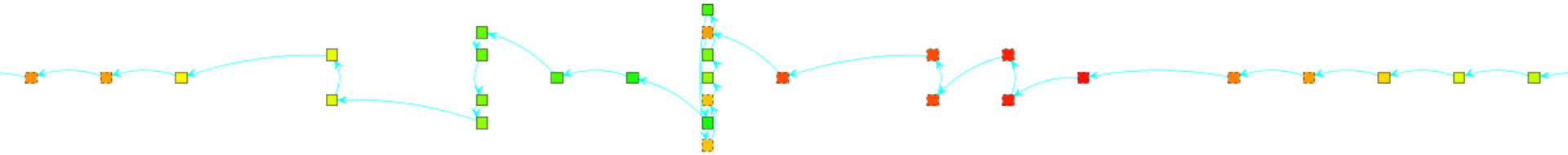


Car Tutorial

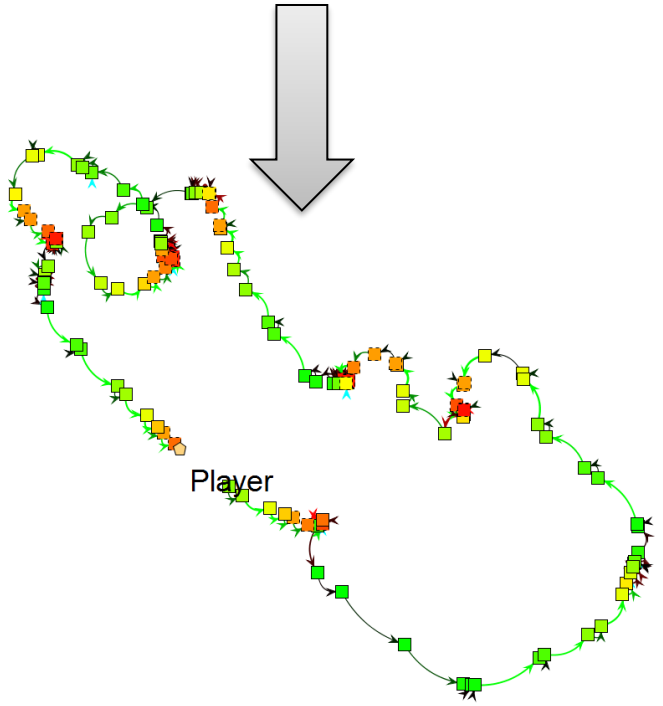
https://www.assetstore.unity3d.com/en/#!/content/10

- E para deixar o grafo assim?

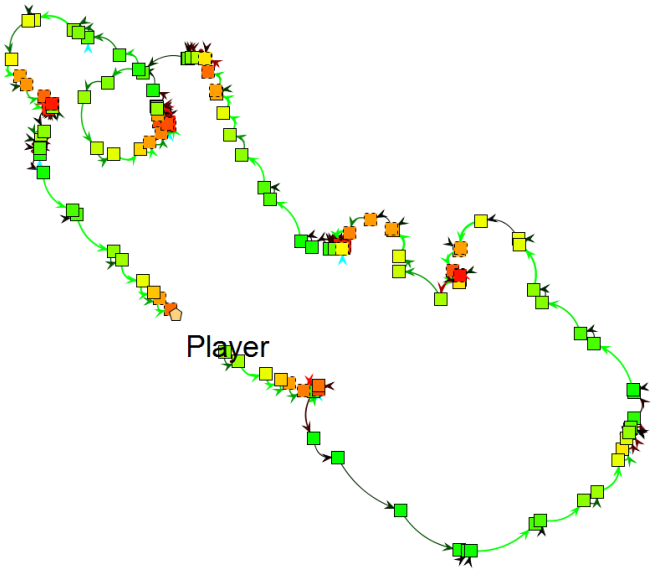
Referenciamento Espacial



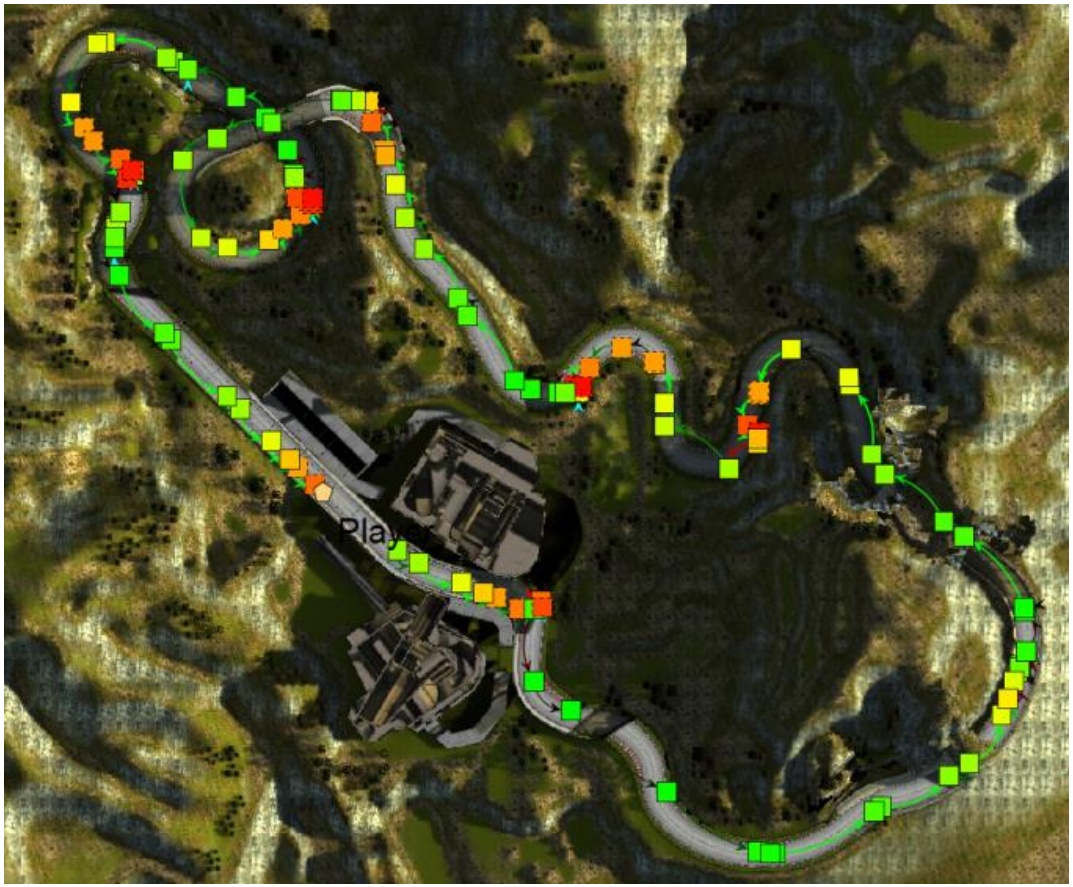
Grafo de Proveniência

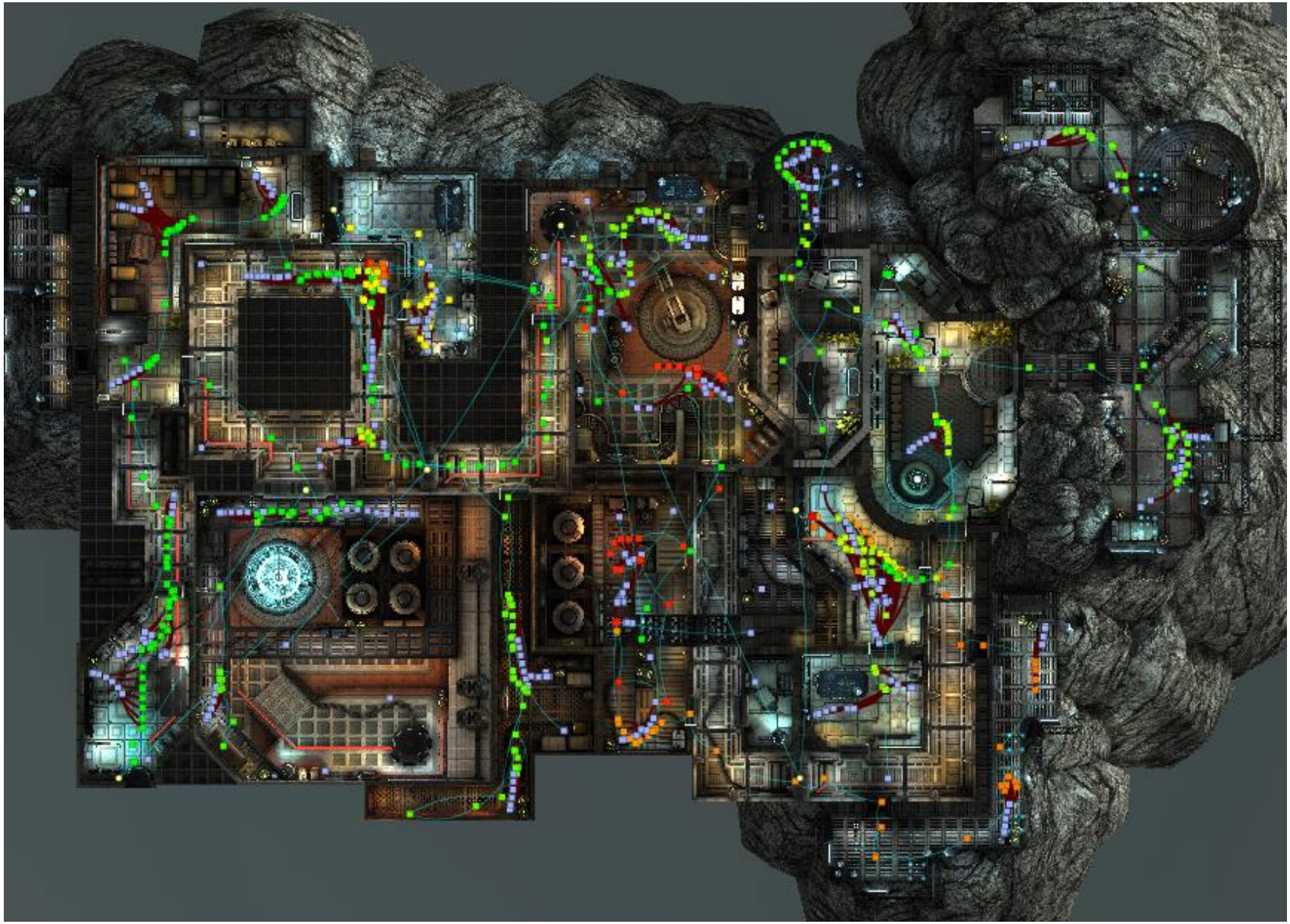


Referenciamento Espacial



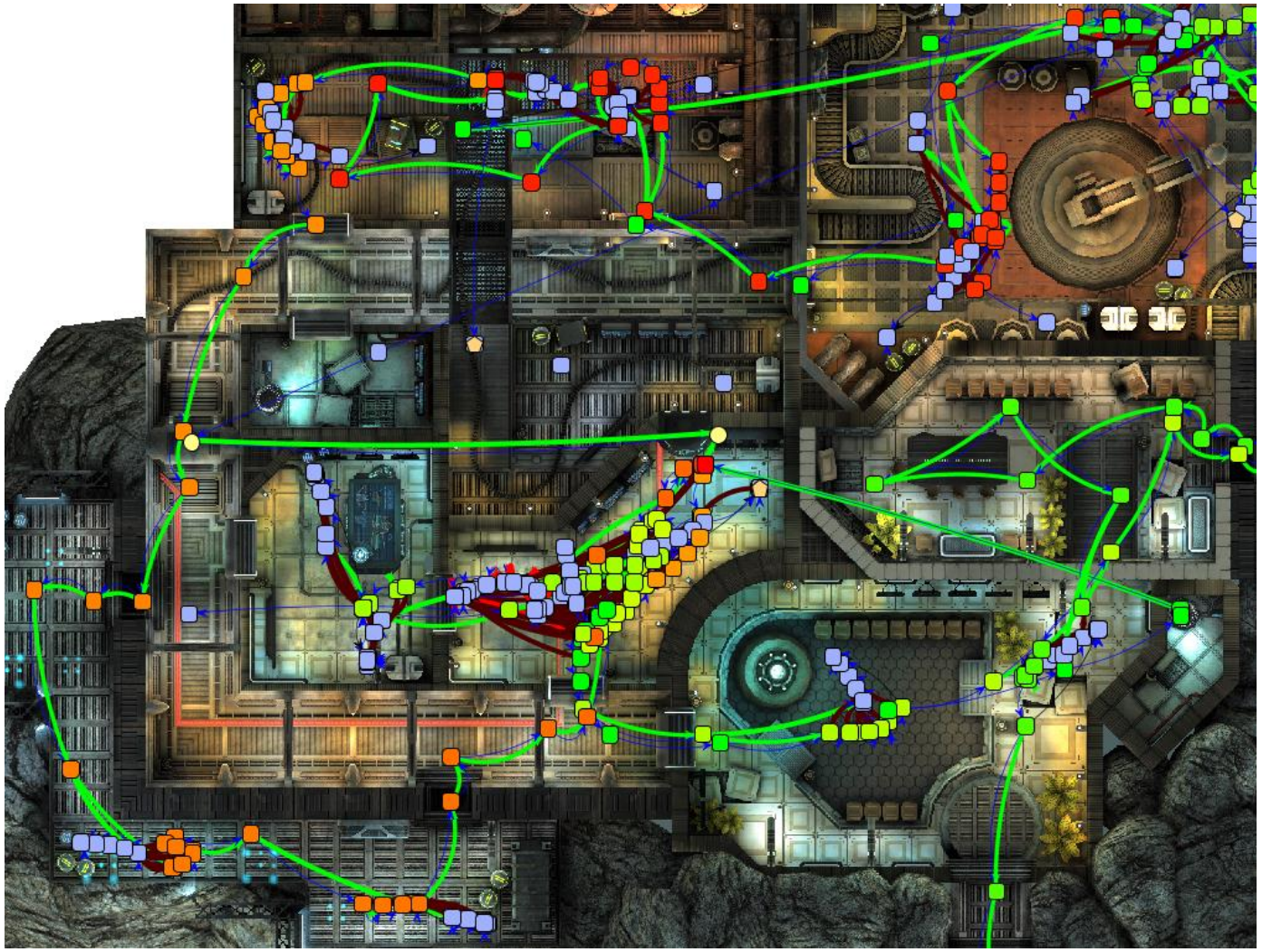
Referenciamento Espacial



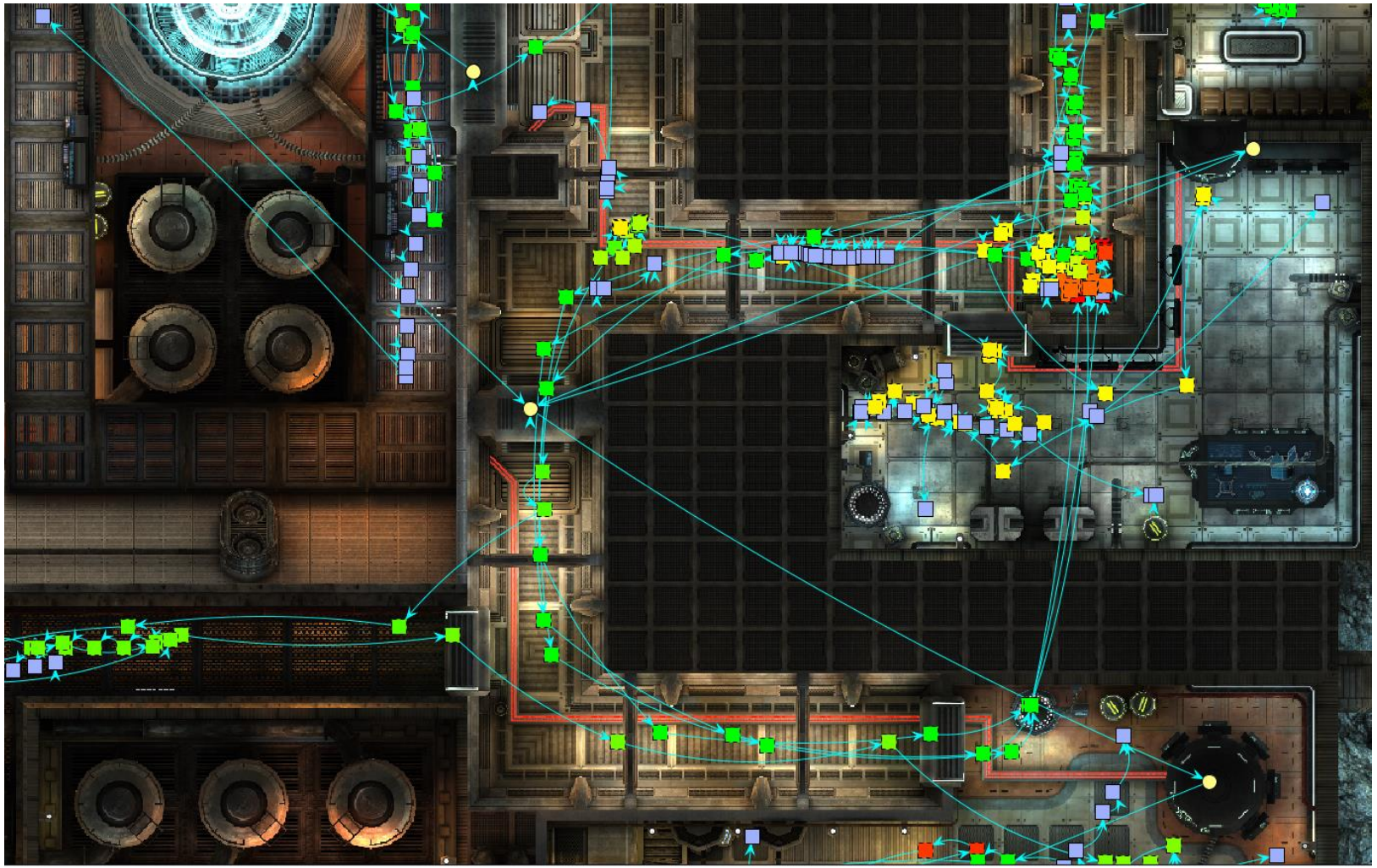


Angry Bots

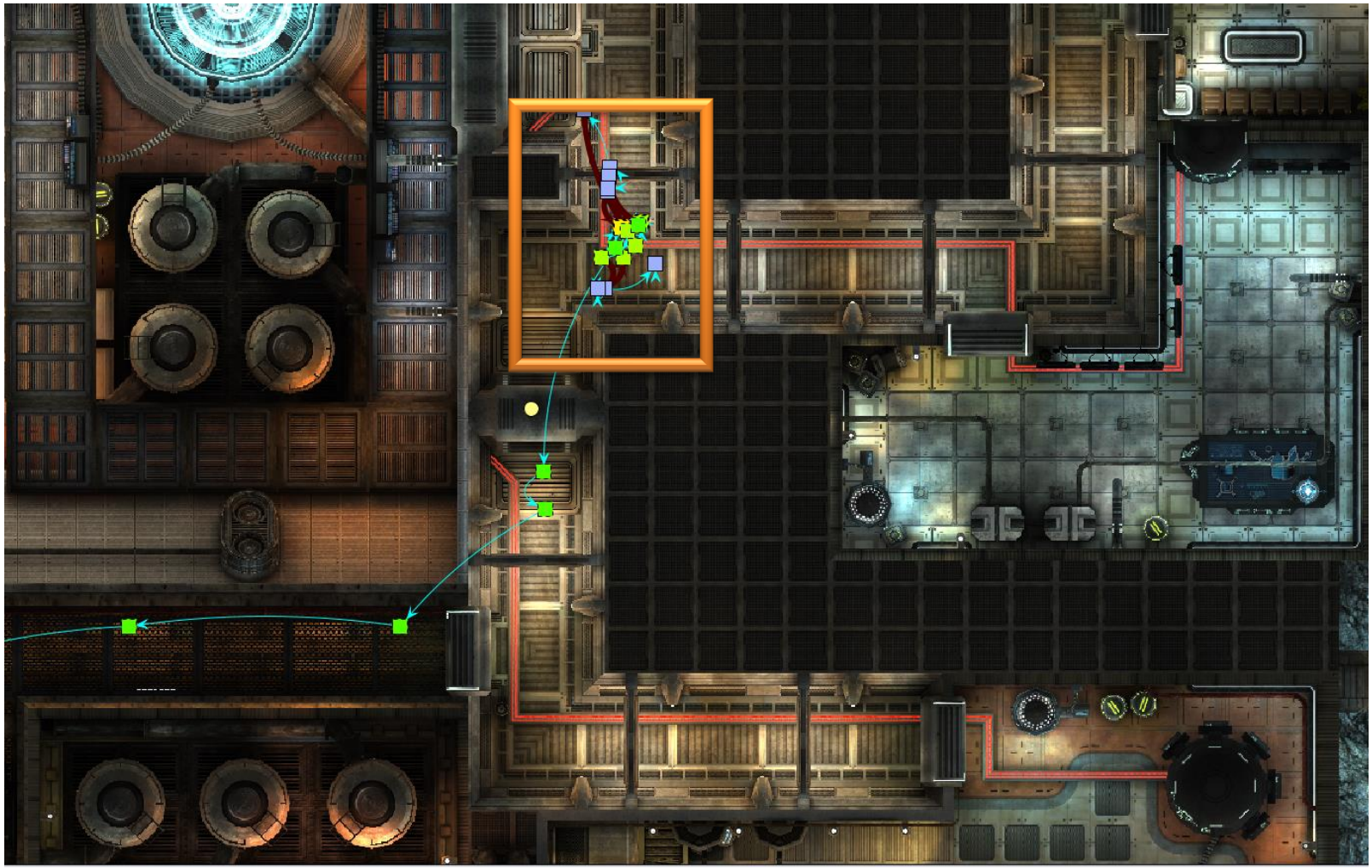
https://www.assetstore.unity3d.com/en/#!/content/12175



Zoom



1st Combat



2nd Combat

3rd Combat

Telemetria de Jogos
Proveniência
PinG
PinGU
Visualização
Considerações Finais

CONSIDERAÇÕES FINAIS

Conclusão

- Telemetria de jogos
  - Exploração de dados
  - Análise visual
  - Mineração
  - etc

- Captura de relações causais
  - I.E., Influencias nas ações

Redes Neurais

- Grafos de redes neurais
  - Extensão de modelos de redes neurais
  - Importância da estrutura topológica dos dados
- Grafos de Proveniências
  - Aprendizado supervisionado
  - Classificação (nós e grafos)
  - Subgraph matching
  - Predição

Imitation Learning

"Refere-se à aquisição por parte do agente de habilidades ou comportamentos, observando um professor que demonstra uma determinada tarefa"

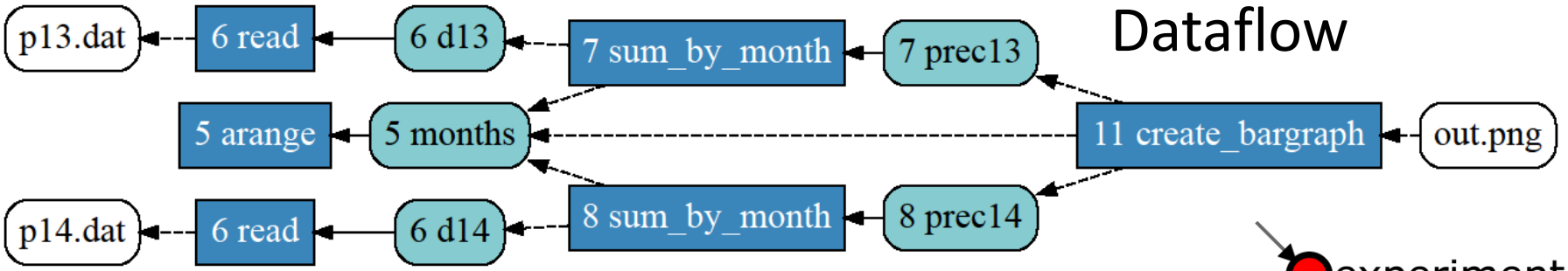
- Aprender como os jogadores jogam
  - Desenvolver agentes autônomos

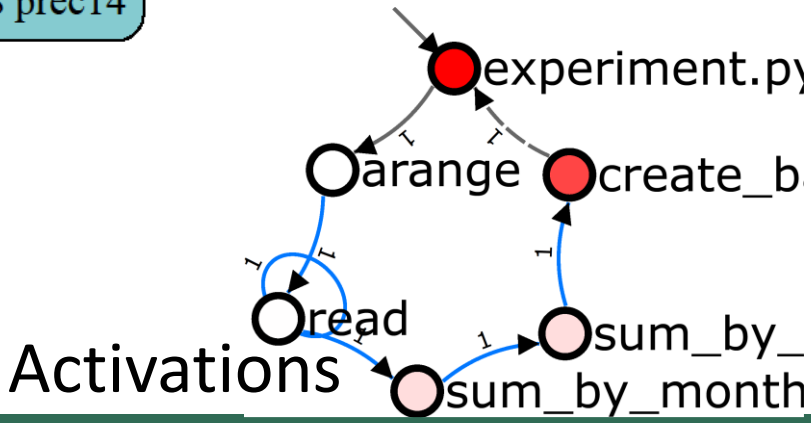
Balanceamento de Jogos

- Apreender ações e comportamentos
- Adaptar o jogo de acordo com o jogador

noWorkflow

- Coleta **transparentemente** a proveniência em Python
  - Nenhuma alteração necessária nos scripts
- Permitir usuários **analisar** e **gerenciar** a proveniência





Proveniência em Jogos

PinGU
http://gems-uff.github.io/ping/

Prov Viewer
http://gems-uff.github.io/prov-viewer/

Troy C. Kohwalter,

Instituto de Computação – Universidade Federal Fluminense (UFF)

Instituto de Computação

NYU TANDON SCHOOL OF ENGINEERING

MEDIALAB

ESI Laboratório de Engenharia de Sistemas e Informação