Engineering emergence: applied theory for game design
Dormans, J.

Citation for published version (APA):
Dormans, J. (2012). Engineering emergence: applied theory for game design Amsterdam: Creative Commons

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
B.1 Static Engine

Type
Engine

Intent
Produce a steady flow of resources over time for players to consume or to collect while playing the game.

Motivation
A static engine creates a steady flow of resources that never dries up.

Applicability
Use a static engine when you want to limit players’ actions without complicating the design. A static engine forces players to think how they are going to spend their resources without much need for long-term planning.

Structure

A static engine must provide players with some options to spend the resources on. A static engine with only one option to spend the resources on is of little use.
Participants

- **Energy** - the resource produced by the static engine.
- **Actions** - a number of game actions the player can spend energy on.

Consequences

The production rate of a static engine does not change. Therefore the effects of the engine on game balance are very predictable. A static engine can only be the cause of imbalance when the production rate of a static engine is not the same for all the players.

A static engine generally does not inspire long term strategies: collecting resources from a static engine, if possible at all, is going to be quite obvious.

Implementation

Normally it is very simple to implement a static engine: a simple source that produces the energy will suffice. It is possible to add multiple steps in the energy production, but in general this will add little to the game.

A static engine can be made unpredictable by using some form of randomness in the production. An unpredictable static engine will force the player to prepare for periods of fewer resources and reward players that make plans that can withstand bad luck. The easiest way to create an unpredictable static engine is to use randomness to vary the output of resources or the time between moments of production, but skill or multiplayer dynamics could work as well.

The outcome of random production rates can be, but does not need to be, the same for every player. By using an unpredictable static engine that generates the same resources for all players the luck factor is evened out without affecting the unpredictability. This puts more emphasis on the planning and timing the pattern introduces. An example would be a game where all players secretly decide how many resources all players can get. The lowest number will be the number of resources to enter play for everyone, while the players who offered the lowest can act first. This would automatically set up some feedback from the games current state to this mechanism.

Examples

In many turn based games the limited number of actions a player can perform each turn can be considered a static engine. In this case the focus of the game is the choice of actions and generally players cannot save actions for later turns.

The fantasy board game *Descent: Journeys in the Dark* uses this mechanism. Players can choose between one of four actions for their hero every turn: move twice, attack twice, move once and attack once, or, move or attack and prepare a special action (see figure B.1).

The energy produced by the spacecrafts in the *Star Wars: X-Wing Alliance* is an example of a static engine. The energy can be diverted to boost
Figure B.1: Distribution of action points in the board game Descent: Journeys in the Dark.

The player’s shields, speed and lasers. This is a vital strategic decision in the game and the energy allocation can be changed at any moment. The amount of energy generated every second is the same for all spacecraft of the same type (see figure B.2).

Fearsome Floors employs an interesting variation of the static engine. In this game the player controls a team of three or four characters that need to escape the game board. Each turn each character has a limited number of action points which he can use to move and push object around. The number of action points oscillates between two values (that for all character add up to seven). This makes the game surprisingly less predictable, and requires the player to plan ahead at least one turn.

In Up the River players roll a die each turn and can move one of their ships as many places closer to the harbor. Because the player controls three ships and ships that are not moved eventually fall of the board, the decision of what ship to move is not as straightforward as it might seem at first glance. Elements on the board add extra complications that further emphasize planning and looking ahead.

In Backgammon dice determine how far the player can move up to two of his playing pieces, or up to four if he rolled a double. As the player has fifteen playing pieces in total there are many options. In important strategy in Backgammon is to move the playing pieces in such way that they are safe from the opponent and give the player as many possible options for the next turn.

The Game of Goose or Snakes and Ladders are poor implementations of an unpredictable static engine as both games lack multiple actions: the player

Figure B.2: Distribution of energy in Star Wars: X-Wing Alliance.
does not have any choice and is left to the whims of chance.

Related Patterns
A (weak) static engine can prevent deadlocks in a converter engine.

B.2 Dynamic Engine

Type
Engine

Intent
A source produces an adjustable flow of resources. Players can invest resources to improve the flow.

Motivation
A dynamic engine produces a steady flow of resources and opens the possibility for long term investment by allowing the player to spend resources to improve the production. The core of a dynamic engine is a positive constructive feedback loop.

Applicability
Use a dynamic engine when you want to introduce a trade-off between long term investment and short-term gains.

Structure

Participants
- **Energy** - the resource produced by the dynamic engine.
- **Upgrades** - a resource that affects the production rate of energy.
- **Actions** - a number of game actions the player can spend on, including:
• **Invest** - an action that creates upgrades.

**Collaborations**

The dynamic engine produces energy that is consumed by a number of actions. One action (invest) produces upgrades that improves the energy output of the dynamic engine. A dynamic engine allows two different types of upgrades a player can invest to improve:

- The interval at which energy is produced.
- The number of energy tokens generated each time.

The differences between the two are subtle, with the first producing a more steady flow than the second, which will lead to bursts of energy.

**Consequences**

A dynamic engine creates a powerful positive constructive feedback loop that probably needs to be balanced by some pattern implementing negative feedback, such as any form of friction, or use escalation to create challenges of increasing difficulties.

When using a dynamic engine one must be careful not to create a dominant strategy either by favoring the long term strategy, or by making the costs for the long term strategy to high.

A dynamic engine generates a distinct gameplay signature. A game that consist of little more than a dynamic engine will cause the players to invest at first, appearing to make little progress. After a certain point, the player will start to make progress and needs to try and do so at the quickest possible pace. The charts these patterns typically generate quite clearly show this behavior. This effect of play is clearly visible in MONOPOLY (see figure B.3, see also section 4.8).

**Implementation**

The chance of building a dominant strategy that favors either long-term or short-term investment is lessened when some sort of randomness is introduced in the

![Figure B.3: The gameplay signature of a dynamic engine.](image-url)
dynamic engine. However, the positive feedback loop that exists in an unpredictable dynamic engine will amplify the luck a player has in the beginning of the game, which might result in too much randomness quickly.

The outcome of random production rates can be, but does not need to be, the same for every player. By using an unpredictable dynamic engine that generates the same resources for all players the luck factor is lessened without affecting the unpredictability. This puts more emphasis on the player’s chosen strategy.

Some dynamic engines allow the player to convert upgrades back into energy, usually against a lower rate than the original investment. When upgrades are expensive, and the player frequently needs large amounts of energy this becomes a viable option.

Examples

In StarCraft one of the abilities of SUV units is to harvest minerals which can be spend on creating more SUV units to increase the mineral harvest (see figure B.4). In its essence this is a dynamic engine that propels the game (although in StarCraft the number of minerals is limited, and SUV units can be killed by enemies). It immediately offers the player a long term option (investing in many SUV units) and a short term option (investing in military units to attack enemies quickly or respond to immediate threats).

Settlers of Catan has at its core a dynamic engine affected by chance (see figure B.5). The roll of the dice which game board tiles will produce resources at the start of each player’s turn. The more villages the player builds the more chance that the player will receive resources every turn. The player can also upgrade villages into cities which doubles the resource output of each tile. Settlers of Catan gets around the typical signature a dynamic engine creates by allowing different types of invest actions, and using a measure of upgrades instead of energy to determine the winner.

Related Patterns

The dynamic friction and attrition patterns are good ways to balance a dynamic engine.
B.3 Converter Engine

Type
Engine

Intent
Two converters set up in a loop create a surplus of resources that can be used elsewhere in the game.

Motivation
Two resources that can be converted into each other fuel a feedback loop that produces a surplus of resources. The converter engine is a more complicated mechanism than most other engines, but also offers more opportunities to improve the engine. As a result a converter engine is nearly always dynamic.

Applicability
- Use the converter engine when you want to create a delicate mechanism to provide the player with resources. It increases the difficulty of the game as the strength and the required investment of the feedback loop are more difficult to assess.
- Use the converter engine when you need multiple options and mechanics to tune the signature of the feedback loop that drives the engine, and the thereby the stream of resources that flow into the game.

Figure B.5: The production mechanism in Settlers of Catan.
Structure

Participants

- Two resources: energy and fuel.
- A converter that changes fuel into energy.
- A converter that changes energy into fuel.
- Actions that consume energy.

Collaborations

The converters change energy into fuel and fuel into energy. Normally the player ends up with more energy than the player started with.

Consequences

A converter engine introduces the chances of a deadlock, when both resources dry out, the engine stops working. Players run the risk of creating deadlocks themselves by forgetting to invest energy to get new fuel. Combine a converter engine with a weak static engine to prevent this from happening.

A converter engine requires more work from the player, especially when the converters need to be operated manually.

As with dynamic engines, a positive feedback loop drives a converter engine, in most cases this feedback loop needs to be balanced by applying some sort of friction.

Implementation

The number of steps involved in the feedback loop of a converter engine for a large part determines its difficulty to operate efficiently: more steps increase the difficulty, fewer steps reduce the difficulty. At the same time the number of steps also positively affects the number of opportunities for tuning or building the engine.

With too few steps in the system, the advantages of the converter engine are limited and one might consider replacing it with a dynamic engine. Too many
Figure B.6: The production mechanism in POWER GRID. The converter engine is printed bolder.

Figure B.7: Elaboration of a converter used in POWER GRID.

steps might result in an engine that is cumbersome to operate and/or maintain, especially in a board game where the different elements of the engine usually cannot be automated.

It is possible to create an unpredictable converter engine by introducing randomness, multiplayer dynamics or skill into the feedback loop. This complicates the converter engine further and often increases the chances of deadlocks.

Many implementations of the converter engine pattern put a limiter gate somewhere in the cycle in order to keep the positive engine under control and to keep the engine from producing too much energy. For example, if the number of fuel resources that can be converted each turn is limited the maximum rate at which the engine can run is capped.

Examples

A converter engine is at the heart of POWER GRID (see figure B.6). Although one of the converters is replaced by a slightly more complex construction (see figure B.7). The players spend money to buy fuel from a market, and use that fuel to generate money in power plants.¹ The surplus money is invested in more efficient power plants and connecting more cities to the player’s power network. The converter engine is limited: the player can only earn money for every connected city, which effectively caps the energy output during a turn, and provides another opportunity for dynamic engine building. POWER GRID also has a weak static engine to prevent deadlocks: the player will collect a little

¹The fiction of the game is that players generate and sell electricity, but as electricity is not a tangible resource in the game it is left out of this discussion.
amount money during a turn even if the player failed to generate money through power plants. The converter engine of POWER GRID is slightly unpredictable as players can drive up to price of fuel by stockpiling it, which acts as a stopping mechanism at the same time.

The eighties computer space-trading game ELITE features an economy that occasionally acts as a converter engine. In ELITE every planet has its own market selling and buying various trade goods. Occasionally players will discover a lucrative trade route where they can buy one trade good at planet A and sell it at a profit planet B, and return with another good which is in high demand on planet A again (see figure B.8). Sometimes these routes involve three or more planets. Essentially such a route is converter engine. It is limited by the cargo capacity of the player’s ship, which for a price can be extended. Other properties of the player’s ship might also affect the effectiveness of the converter engine: the ship’s “hyperspeed” range, and also its capabilities (or cost) to survive a voyage through hostile territories all affect the profitability of particular trade routes. Eventually trade routes become less profitable as the player’s efforts reduce the demand, and thus the price, for certain goods over time (a mechanism that is omitted from the diagram).

**Related Patterns**

A converter engine is well suited to be combined with the engine building pattern as it has many opportunities to change settings of the engine: the conversion rate of two converters and possibly the setting of a limiter.

A converter engine is best balanced by introducing some sort of friction or some other form of negative feedback.
B.4 Engine Building

Intent
A significant portion of gameplay is dedicated to building up and tuning an engine to create a steady flow of resources.

Motivation
A dynamic engine, converter engine or a combination of different engines form a complex and dynamic core of the game. The game includes at least one, but preferably multiple, mechanics to improve the engine. These mechanics can involve multiple steps. It is important that assessing the current state of the engine is no trivial task.

Applicability
Use engine building when you want to create a game that:

- has a strong focus on building and construction.
- focuses on long-term strategy and planning.

Structure

The structure of the Core Engine is an example. There is no fixed way of building the engine. Engine building only requires that several building mechanics operate on the engine and that the engine produces energy.

Participants

- The core engine usually is a complex construction combining multiple engine types.
- At least one, but usually multiple building mechanisms to improve the core engine.
- Energy is the main resource produced by the core engine.
Collaborations

Building mechanisms increase the output of the engine. If energy is required to activate building mechanisms then a positive, constructive feedback loop is created.

Consequences

Engine Building increases the difficulty of a game, it is best suited for lower paced games as it involves planning and strategic decisions.

Implementation

Including some form of unpredictability is a good way to increase difficulty, generate varied gameplay and avoid dominant strategies. Engine building offers many opportunities create unpredictability as the core engine tends to consist of many mechanisms. The complexity of the core engine itself usually also causes some unpredictability.

When using the engine building pattern with feedback it is important to make sure the positive, constructive feedback is not too strong and not too fast. In general, you want to spread out the process of engine building over the entire game.

An engine building pattern operates without feedback when energy is not required to activate building mechanisms. This can be viable structure when the engine produces different types of energy that affect the game differently, and allows the players to follow strategy that favor particular forms of energy above others. However, it usually does require that the activation of building mechanisms is restricted in some way.

Examples

SimCity is a good example of engine building. The energy in SimCity is money which is used to activate most building mechanisms, which in the game consists of preparing building sites, zoning, building infrastructure, building special buildings, and demolition. The core engine of SimCity is quite complex with many internal resources such as people, job vacancies, power, transportation capacity, and three different types of zones. Feedback loops within the engine cause all sorts of friction and effectively balance the main positive feedback loop, up to the point that the engine can more or less collapse if the player is not careful and manages the engine well.

In the board game Puerto Rico each player builds up a New World colony. The colony generates different types of resources that can be reinvested into the colony or converted into victory points. The core engine includes many elements and resources such as plantations, buildings, colonists, money and a selection of different crops. Puerto Rico is a multiplayer game in which they players compete for a limited number of positions which allow different actions
to improve the engine; they compete for different building mechanics. This way
a strong multiplayer dynamic is created that contributes much of its gameplay.
Most real-time strategy games follow a complex engine building pattern com-
bined with the attrition pattern.

Settlers of Catan follows an engine building structure that uses feedback
and multiple types of energy. In this particular case the energy produced by the
engine is never used outside the engine and its building mechanisms. This works
in Settlers of Catan because the objective of the game is to score a number
of points that are directly derived from the number of upgrades to the engine.

Related Patterns

Applying multiple feedback to the building mechanisms is a good way to increase
the difficulty of the engine building pattern.

All friction patterns are suitable to balance the possible positive feedback
that is created by an implementation of engine building that consumes energy
to activate building mechanisms.

B.5 Static Friction

Type

Friction

Intent

A drain automatically consumes resources produced by the player.

Motivation

The static friction pattern counters a production mechanism by periodically
consuming resources. The rate of consumption can be constant or subjected to
randomness.

Applicability

• Use static friction to create a mechanism that counters production, but
  which can eventually be overcome by the players.

• Use static friction to exaggerate the long-term benefits from investing in
  upgrades for a dynamic engine.
Structure

![Diagram showing energy flow from production mechanism to action through energy]

Participants

- A resource: **energy**.
- A **static drain** that consumes energy.
- A **production mechanism** that produces energy.
- Other **actions** that consume energy.

Collaborations

The production mechanism produces energy players need to use to perform actions. The static drain consumes energy outside players direct control.

Consequences

The static friction pattern is a relative simple way to counter positive feedback created by engine patterns. Although it tends to accentuate the typical gameplay signature of the dynamic engine because it makes upgrades of the static engine more valuable as the friction is not affected by the number of upgrades, current energy levels, or players’ progress.

Implementation

An important consideration when implementing static friction is whether the consumption rate is constant or subjected to some sort of randomness. Constant static friction is the easiest to understand and most predictable, whereas random static friction can cause more noise in the dynamic behavior in the game. The latter can be good alternative to using randomness in the production mechanism. The frequency of the friction is another consideration: when the feedback is applied at short intervals the overall system will be more stable than when the feedback is applied at long or irregular intervals, which might cause periodic behavior in the system. In general, the effects of a gradual loss of energy on the dynamic behavior of the system is less than the same amount of energy is lost periodically.
Examples

In the Roman city building game CAESAR III the player needs to pay tributes to the emperor at particular moments during each mission. The schedule of the tributes is fixed for each mission and not affected by the player’s performance. In effect, they cause uses a very infrequent, and high form of static friction, that causes a huge tremor in the game’s internal economy. See section 6.3 for a more detailed discussion of this game.

The dynamic engine in MONOPOLY is countered by different types of friction, including static friction (see figure B.9). The main mechanism that implements static friction is the chance cards through which the player infrequently looses money. Although some of these cards take into account the player’s property, most of them do not. One could also argue that paying rent to other players is also a form of static friction, as the frequency and severity is far beyond the direct control of the player. But, the rate of the friction does change over time and players have some indirect effect on it: when a player is doing well, chances are that the opponents are not, which negatively impacts this friction. The diagram in figure B.9 does not show this as it is made from the scope of an individual player. This type of friction is an example of the attrition pattern.

Related Patterns

As static friction exaggerates long-term investments it is best suited to be used with combination with a static engine, converter engine, or an engine building pattern.

B.6 Dynamic Friction

Type

Friction
Intent
A drain automatically consumes resources produced by the player, the consumption rate is affected by the state of other elements in the game.

Motivation
Dynamic friction counteracts production but adapts to the performance of the player. Dynamic friction is a very typical application of negative feedback in a game.

Applicability
- Use dynamic friction to balance games where resources are produced too fast.
- Use dynamic friction to create a mechanism that counters production that automatically scales with players’ progress or power.
- Use slow dynamic friction to reduce the effectiveness of long term strategies created by a dynamic engine in favor for short term strategies.

Structure

Participants
- A resource: energy.
- A dynamic drain that consumes energy.
• A **production mechanism** that produces energy.

• Other **actions** that consume energy.

**Collaborations**

The production mechanism produces energy players need to use to perform actions. The dynamic drain consumes energy outside players direct control, but affected by the state of at least one other element in the game system.

**Consequences**

Dynamic friction is a good way to counter positive feedback created by engine patterns. Dynamic friction adds a negative feedback loop to the game system.

**Implementation**

There are several ways of implementing dynamic feedback. An important consideration is the choice of the element that causes the consumption rate to change. In general, this can either be the energy itself, the number of upgrades to a dynamic engine or a converter engine, or the player’s progress towards a goal. When energy is the cause for the friction to change, the negative feedback tends to be fast, when progress or production power is the cause the feedback is more indirect, and probably slower.

When dynamic friction is used to counter a positive feedback loop it is important to consider the difference in characteristics of the positive feedback loop and the negative feedback loop implemented through the dynamic friction. When the characteristics are similar (equally fast, equally durable, etc.), the effect is far more stable that when the differences are large. For example, when a slow and durable dynamic friction is acting against a fast but non-durable positive feedback that initially yields a good return, players will initially make a lot of progress, but might suffer in the long run. Fast positive feedback and low negative feedback seems to be the most frequently encountered combination.

**Examples**

Dynamic friction is used the city production mechanism in **CIVILIZATION** (see figure B.10). In this game the player builds cities to produce food, shields and trade. As cities grow they need more and more food for their own population. Players have some control over how much food is produced compared to other resources, but are limited in their options by the surrounding terrain. By choosing to produce a lot of food cities initially produce fewer other resources, but grow faster to great potential. Fast growth creates a problem however, the happiness rating a city must stay equal to or higher than half the population, else the production stops because of civil unrest. Initially a city has a happiness value of 2. Players can create more happiness by building special buildings, or by converting trade into culture. Both ways cause more dynamic friction with different
Figure B.10: The city economy of Civilization. Dynamic friction is printed bolder. The player can freely adjust the culture and research settings in order to control unrest and research production. These settings are global and affect all cities equally.

profiles on the production process; the first is slow, requires a high investment, but is highly durable, and has relative high return, whereas the second is fast, but has a relative low return for its investment.

Related Patterns
Dynamic friction is a good way to balance any pattern that causes positive feedback, and often is part of the multiple feedback pattern.

B.7 Attrition
Type
Friction

Intent
Players actively steal or destroy resources of other players that they need for other actions in the game.
Also Known As
Multiplayer, destructive feedback

Motivation
By allowing player to directly steal or destroy each others resources, players can eliminate each other in a struggle for dominance.

Applicability
- Use attrition to allow direct and strategic interaction between multiple players.
- Use attrition to introduce feedback into a system which nature is determined by the strategic preferences or whims of the players.

Structure

![Diagram of the structure]

Participants
- Multiple players that have the same (or similar) mechanics and options.
- A resource used by all players: energy to perform actions.
- A special attack action that triggers or activates a drain of other player’s energy.

Collaborations
By performing attack actions players can drain each other’s energy. Attacking typically costs some sort of energy too, or at least attacking players can spend less time on other actions. The balance between the attack costs, how much energy players loose to an attack, and how beneficial the other actions are, determine the effectiveness the attack and the pattern.

Consequences
Attrition introduces a lot of dynamism into a system as players directly control the measure of the destructive force applied to each other. Often this introduces destructive feedback as the current state of a player will cause reactions by other
players. Depending on the nature of the winning conditions and the current state of the game this feedback might be negative when it stimulates players to act and conspire against the leader, but it also might cause positive feedback when players are stimulated to attack and eliminate weaker players.

Implementation

For attrition to work it is best that players must invest some sort of resource into attack that could also be spend otherwise. Without this investment attrition simply becomes race to destroy each other with little or no strategic choices. Although this might work when there are multiple players are involved and the game facilitates social interaction between players. In this case the players need to chose whom to attack.

It is quite common to implement attrition using two resources to represent life and energy. In this case players use energy to perform actions and loose when they run out of life. When using these two resources it is important that they are somehow related, often players are allowed to spend energy to gain more life. Sometimes the relation between life and energy is implicit, for example when a player must choose between spending energy or gaining life, there is an implicit link between the two as players generally cannot do both at the same time.

In a two player version of attrition the game must include other actions, and games for more than two players often also allow other actions to be performed by the player. Most of the time a these actions constitute some sort of production mechanism for energy, increases the effectiveness the players defensive or offensive capabilities. Most real-time-strategy games include all these options, often with multiple variants for each.

The winning conditions and effects of eliminating another player have a big impact on the attrition pattern. The winning condition does not need be the elimination of players, players might score points, or reach a particular goal outside the attrition pattern which automatically widens the number of strategies involved. When there is a bonus to attacking or eliminating players, the pattern can be made to stimulate weaker players.

Examples

Almost all strategy game implement some sort of attrition as it is often an important goal to eliminate other players in this type of games. The SimWAR example, discussed in Chapter 4, provides a good example.

The trading card game MAGIC THE GATHERING implements a decorated version of the attrition pattern. Figure figure B.11 presents this implementation, although it show the details from the perspective of a single player only. In MAGIC THE GATHERING, players can play one card every turn, these cards allow players to add lands, summon creatures or cast spells to heal, or deal direct damage to their opponent or their opponent’s creatures. But all actions, except playing lands costs mana, the more mana players have the more they can spend each turn and the more powerful actions they can play. Creatures will fight
other creatures and when there are no more enemy creatures they will damage the opponent directly. Players that loose all their life points are eliminated from the game. Magic the Gathering is an example of a game that implements attrition using separate resources for life and energy (in this case life and mana). The different options illustrate how attrition can work differently: direct damage activates a trigger briefly. As its name implies, it is fast and direct. On the other hand, summoning creatures activates a permanent drain on the opponent’s creatures and life. The effects usually are not as powerful as direct damage, but as they accumulate over time they can be quite devastating. What options are available to a player and how powerful these options are exactly, is determined by the cards in the player’s hand. As players build their own decks from a large collection of cards, deck building is an important aspect of Magic the Gathering.

The most obvious way to implement attrition is in a symmetrical game. However, many single-player games and even certain types of multiplayer games use ‘a-symmetrical’ attrition. An example of ‘a-symmetrical’ attrition can be found in the board game Space Hulk where one player, controlling the a handful of ‘space marines’, tries to accomplish a mission while the other player, controlling an unlimited supply of alien ‘Genestealers’ tries to prevent that. The genestealer player tries to reduce the number of space marines to stop them from accomplishing their goals, they win when they have destroyed enough space marines. On the other hand, the space marines usually cannot win by destroying a certain number of genestealers, but they must keep the number of genestealers under control in order to survive, as the genestealers become more effective as their numbers grow. Figure B.12 is a rough illustration of these mechanics in Space Hulk.
Related Patterns

Attrition works well with any sort of engine pattern. Trade can be used as an alternative form of multiplayer feedback that is constructive instead of destructive, and nearly always negative.

B.8 Stopping Mechanism

Type
Friction

Intent
Reduce the effectiveness of a mechanism every time it is activated.

Motivation
In order to prevent a player from abusing a powerful mechanism its effectiveness is reduced every time it is used. In some cases the stopping mechanism is permanent, but usually its not.

Applicability
Use stopping mechanism to:
- prevent players from abusing particular actions.
- counter dominant strategies
- reduce the effectiveness of a positive feedback mechanism.
Appendix B | Machinations Design Patterns

Structure

- An **action** that might produce some sort of **output**.
- A resource **energy** that is required for the action.
- The **stopping mechanism** that increases the energy cost or reduces the output of the action.

Collaborations

For a stopping mechanism to work it the action must have an energy cost, an output, or both. The stopping mechanism reduces the effectiveness of an action mechanism every time it is activated by increasing the energy costs or reducing the output.

Consequences

Using a stopping mechanism can reduce the effect a positive feedback loop considerably and even make its return insufficient.

Implementation

When implementing a stopping mechanism is important to consider whether or not to make the effects permanent or not. When the accumulated output is used to measure the strength of the stopping mechanism the effects are not permanent, in that case it requires players to alternate frequently between creating the output and using the output in other actions.

A stopping mechanism can apply to each player individually or can affect multiple players equally. In the latter case, players will reward players that use the action before other players do. This means that the stopping mechanism can create a form of feedback depending on whether leading or trailing players are likely to act first.

Examples

The price mechanic of the fuel market in **Power Grid** involves a stopping mechanism (see figure B.13). In **Power Grid** players use money to buy fuel
Figure B.13: The stopping mechanism in Power Grid drives up the price for fuel and causes negative feedback, especially for leading players.

and burn fuel to generate money. This positive feedback loop is counteracted by the fact that buying a lot of fuel actually drives up the price for all players. As in Power Grid the leading player acts last, this stopping mechanism causes more negative feedback for the leading player.

Related Patterns
A stopping mechanism is often used as a way to implement multiple feedback.

B.9 Multiple Feedback

Intent
A single gameplay mechanism feeds into multiple feedback mechanisms, each with a different signature.

Motivation
An player action activates multiple feedback loops at the same time. Some feedback loops will be more obvious than others. This creates a situation where the exact outcome or success of an action might be predictable on the short term, but can have unexpected result on the long run.

Applicability
Use multiple feedback to:
- increase a game’s difficulty.
- emphasize the player’s ability to read the current game state.
**Structure**

In the example structure above there are two feedback mechanisms. The action (black) activates one feedback mechanism (red) which is positive, fast and direct, but it also activates a secondary feedback mechanism (blue) which is negative, slow and indirect. This illustrates just one way of setting up multiple feedback loops. There are many more.

**Participants**

- An **action** that can be activated by the player.
- Multiple **feedback mechanisms** that are activated by the action.

**Collaborations**

The action activates multiple feedback mechanisms that ultimately feedback into the action.

**Consequences**

For the player multiple feedback loops are more difficult to understand that single feedback loops. As a result using this pattern makes a game more difficult.

If the feedback loops the action activates can have dynamic signatures that change during play (which they often have) it is very important for the player to be able to read the current signature, as their balance might shift considerably during the game.

Finding the right balance between the multiple feedback loops is an important issue in a game that uses this pattern.

**Implementation**

When creating a game with multiple feedback it is very important to make sure that the signatures of the different feedback loops are different. In particular the speed of the feedback needs to vary if this pattern is going to be effective. Alternatively varying the signature of the feedback over time can also work well. To this end adding playing style reinforcements and stopping mechanisms to one or more of the feedback loops is a good design strategy.

The most common combination for multiple feedback seems to be fast, constructive, positive feedback coupled with slow, negative feedback. This creates a trade-off between short term gains and long term disadvantages.
Examples
Attacking in Risk feeds into three positive feedback loops of varying speeds and strengths. The most obviously using armies to capture more lands allows the player to build more armies. A slower form of feedback is realized by the cards: a player that successfully attacks gains a card, certain combination of three cards allow him to gain extra armies. The last type of feedback is created by the capturing of continents. A continent will give a player a number of bonus armies each turn, this is very fast and strong feedback loop, but one that requires a higher investment of the player.

Related Patterns
Playing style reinforcements and stopping mechanisms are good ways to ensure that the signature of the feedback loops an action feeds into change over time.

B.10 Trade
Intent
Allow trade between players to introduce multiplayer dynamics and negative, constructive feedback.

Motivation
Players are allowed to trade important resources. Usually this means that leading players will get tougher deal while trailing players can help each other to catch up. Trade works especially well when the flow of resources is unstable and/or not equal distributed among players.

Applicability
Use trade to:

- introduce multiplayer dynamics to the game.
- introduce negative, constructive feedback.
- introduce a social mechanic that involves players outside their own turns.
Structure

![Diagram of trading mechanism](image)

Participants

- A **trading mechanism** that allows trade of resources.
- Multiple **tradable resources** that can spent in various ways.
- **Actions** that require the use of tradable resources.

Collaborations

The tradable resources can be exchanged by the players using the trading mechanism.

Consequences

Trade introduces negative feedback that does not really slow down the game but usually helps trailing players catch up, (as it is not destructive).

Trade favors players with good social and bartering skills.

Implementation

In board games trade is very easy to implement, the game simply needs to specify how and when players can trade resources. In a multiplayer computer game trade is also easy. However, creating a trading mechanism that involves artificial players, is far from trivial.

To implement a successful trading mechanism, multiple tradable resources are required, and the production rates of these resources must fluctuate or at least be different among players. Trading only works when there is an imbalance in the distribution of resources among the trading parties. It also helps to include many actions that consume the tradable resources and to create actions that consume resources of multiple types, as this further exaggerates the imbalance when players choose different courses of action.

Examples

In *Settlers of Catan* players build up an uncertain dynamic engine: villages and cities that produce the resources used to build more villages and cities. The
randomness of these engines is partly countered by allowing all players to trade resources with the player that is taking his or her turn. The exchange rate is set by in mutual agreement and usually determined by the availability, accessibility of the resource and the position of the player. Players that are in the lead can afford to pay more for their resources. When close to winning players might find it impossible to make a deal.

In Civilization III players can exchange strategic resources, money, and knowledge. This mutually benefits both parties and allows weaker civilizations to catch up rather quickly.

Related Patterns
Attrition can be an alternative source of multiplayer feedback that is not constructive but destructive in nature.

B.11 Escalating Complications
Type
Escalation

Intent
Player progress towards a goal increases the difficulty of further progression.

Motivation
A positive feedback loop on the game’s difficulty makes the game increasingly harder for players as they get closer to achieving their goals. This way the game quickly adapts to the player’s skill level, especially when the good performance allows the player to progress more quickly.

Applicability
Escalating complications is well suited for fast-paced games based on player skill where the game needs to adjust quickly to the player’s skill level.

Structure
Participants

- **Targets** represent unresolved tasks.
- Alternatively, **progress** represents the player’s progress towards a goal.
- A **task** that either reduces the number of targets or produces progress.
- A feedback mechanism that makes the game more difficult as the player progresses towards the goal or reduces the number of targets.

Consequences

Escalating complications is based on a simple positive feedback loop affecting the difficulty of the game. It is mechanism quickly adjusts the difficulty of the game to the skill level of the player. If failure of the task ends the game escalating complication ensures a very quick game.

Implementation

The task in a game that only implements the escalating complication pattern is typically affected by player skill, especially when the escalating complications is makes up the most of a game’s core mechanics. When the task is a random or deterministic mechanic, players will have no control over the game’s progress. Only when the escalating complication pattern is part of a more complex game system, where players have some sort of indirect control over the chance of success, a random or deterministic mechanic becomes viable. Using multiplayer dynamic mechanisms is an option but probably works better in a more complex game system as well.

Examples

**Space Invaders** is a classic example of the escalating complication pattern. In **Space Invaders** the player needs to destroy all invading alien before they can reach the bottom of the screen. Every time the player destroys an alien all other aliens speed up a little, making it more difficult for the player to shoot them.

**Pac-Man** is another example. In **Pac-Man** the task is to eat all the dots in a level, while the chasing ghosts further increase the difficulty of the task and adding to strategic decision to eat an ‘energizer’ to get rid of them.

Related Patterns

By combining static friction or dynamic friction with escalating complications a game can be created that quickly matches its difficulty to the ability of the player.
B.12 Escalating Complexity

Type
Escalation

Intent
Players act against growing complexity, trying to keep the game under control until positive feedback grows to strong and the accumulated complexity makes them loose.

Motivation
Players are tasked to perform an action that grows more complex if the players fail and which complexity contributes to the difficulty of the task. As long as players can keep up with the game they can keep on playing, but once the positive feedback spins out of control, the game ends quickly. As the game progresses the mechanism that creates the complexity gains speeds up ensuring that at one point players can no longer keep up and eventually must lose the game.

Applicability
Use escalating complexity when you aim for an addictive skill based game.

Structure

Participants
- The game produces complexity that must be kept under under a certain limit by the player.
- The task is a player action that reduces complexity.
- Progress mechanism increases the production of complexity over time.
Collaborations

Complexity immediately increases the production of more complexity, creating a strong positive feedback loop that must be kept under control. The player looses when complexity exceeds a particular limit.

Consequences

Given enough skill players can keep up with the creation of complexity for a long time, but when players no longer keeps up complexity spins out of control and the game ends quickly.

Implementation

The task in a game that implements the escalating complexity pattern is typically affected by player skill, especially when escalating complexity makes up most of the game’s core mechanics. When the task is a random or deterministic mechanic, players will have no control over the game’s progress. Only when the escalating complexity pattern is part of a more complex game system, where players have some sort of indirect control over the chance of success, a random or deterministic mechanic becomes viable. Using a multiplayer dynamic task is an option but probably also works better in a more complex game system.

Randomness in the production of complexity creates a game with a varied pace, where players might struggle keeping up with a peek in production before catching some breath when production dies out a little.

There are many ways to implement the progress mechanic, from a simple time based increase of the production of complexity (as is the case in the sample structure above) to complicated constructions that rely on other actions of the player or other player actions. This way it is possible to combine escalating complexity with escalating complication by introducing positive feedback to the progress mechanic as a result of the execution of the task.

Escalating complexity lends itself well to combine with a multiple feedback structure where the complexity feeds into several feedback loops with different signatures. For example, escalating complexity can be partially balanced by having the task feeding into a much slower negative feedback loop on the production of complexity.

Examples

In Tetris a steady flow of falling bricks causes complexity. There is a slight randomness in this production as the different blocks are created all the time. Player need to place the blocks in such way that they fit together closely. When a line is completely filled it disappears making room for new blocks. When players fail to keep up the blocks pile up quickly and they will have less time to place subsequent blocks, this can quickly increase the complexity of the field when players are not careful and causes them to loose the game if the pile of
blocks reach the top of the screen. The game progressively speeds up by making the blocks fall faster and faster, making it more and more difficult.

Related Patterns
Any type of engine pattern can be used to implement the progress mechanism.

B.13 Playing Style Reinforcement

Intent
By applying slow, positive, constructive feedback on player actions, the game gradually adapts to the players preferred playing style.

Also Known As
RPG-elements (as in ‘a game with RPG-elements’)

Motivation
Slow, positive, constructive feedback on players' actions that also have another game effect causes the players' avatar or units to develop over time. As the actions themselves feed back into this mechanism the avatar or units specialize over time, getting better in a particular task. As long as there are multiple viable strategies and specializations, the avatar and the units will, over time, reflect the player’s preferences and style.

Applicability
Use Playing Style Reinforcement when:

- you want players to make a long-term investment in the game that spans multiple sessions.
- you want to reward players for building, planning ahead, and developing personal strategies.
- you want players to grow into a specific role or strategy.
Structure

Participants

- **Actions** players can perform which success is not completely dependent on the players skill.

- A resource **ability** that affects the chance actions succeed and than can grow over time.

- An optional resource **Experience points** that can be used to increase an ability.

Collaborations

Ability affects the success rate of actions.

Performing actions generates experience points or directly improves abilities.

Some games require the action to be successful, while others do not.

Experience points can be spent to improve abilities.

Consequences

Playing style reinforcement works best in games that are played over multiple sessions and over a long time.

Playing style reinforcement only works well when multiple strategies and play styles are viable option in the game. When there is only one, or only a few, every one will go for those options.

Playing style reinforcement can inspire min-maxing behavior with players. With this behavior players will seek the best possible options that will allow them to gain powerful avatars or units as fast as possible. This can happen
when the strength of the feedback is not distributed evenly over all actions and strategies.

Playing style reinforcement favors experienced players over inexperienced players, as the first group will have a clearer view over all the options and long-term consequences of their actions.

Playing style reinforcement rewards the player who can invest the most time in playing the game. In this case, time can compensate for playing skill, which can be a wanted or unwanted side-effect.

It can be very ineffective to change strategies over time in a game with playing style reinforcement, as previous investments in another play style will not be used as efficiently.

**Implementation**

Whether or not to use experience points is an important decision when implementing play style reinforcement. When using experience points there is no direct coupling between growth and action, allowing the player to harvest experience with one strategy to develop the skills to excel in another strategy. On the other hand, if you do not use experience points you have to make sure that the feedback is balanced for the frequency of the actions: actions that are performed more often should have weaker feedback that actions that can be practiced infrequently.

Role-playing games are the quintessential games build around the play style reinforcement pattern. In these games the feedback loops are generally quite slow, and balanced by dynamic friction or a stopping mechanism to make sure avatars do not progress to fast. In fact, most of these games are balanced in such way that progression is initially fast and gradually slows down, usually because the required investment of experience points increases exponentially.

Whether or not the action needs to be executed successfully to generate the feedback is another important design decision and can inspire totally different player behavior. When success is required, the feedback loop gains influence. In that case it is probably best to have tasks difficulty also affect the success of an action, and to challenge the player with tasks of different difficulty levels allowing them to train their avatars. When success is not required, players have more options to improve neglected abilities during later and more difficult stages. However, it might also inspire players to perform a particular action at every conceivable opportunity, which could lead to some unintended unrealistic or comic results, especially when the action involves little risk.

**Examples**

In the board game *Blood Bowl* players coach football teams in a fantasy setting. Individual team members score ‘star player points’ for successful actions: scoring touch downs, throwing complete passes or injuring opposing players. When a team member collected enough star player points he gains new or improved abilities. Many of these increase their ability to score, pass or injure
opponents. Improvements occur only between matches and players build up a team over a long series of matches. **Blood Bowl** facilitates a wide variety of playing styles that generally fall somewhere between two poles: agility play with a strong focus on ball handling and scoring, and strength play with a strong focus on taking out the opposition in order to win the game.

In computer role-playing game *The Elder Scrolls IV: Oblivion* the player avatar’s progress is directly tied to their actions. The avatar’s ability corresponds directly to number of times he performed the associated actions.

In *Civilization III* there are different ways in which a player can win the game. A player reinforces his chosen strategy of military, economic, cultural or scientific dominance (or any combination), by building city improvements and world wonders that favor that strategy. In *Civilization III* several resources take the role of experience points: money and production among them. These resources are not necessarily tied to one particular strategy in the game: money generated by one city can be spent to improve production in another city in the game.

In the board game *Caylus* players occasionally win a privilege. They can choose to use the privilege to gain money, points, building resources, or the opportunity to build new structures. Every time they pick one of them the option is improved (the first time you get 1 point, the second time 2 points, etc). Rules are in place to prevent a player from spending privileges on the same option to often. This is an example of playing style reinforcement without experience points.

**Related Patterns**

When playing style reinforcement depends on the success of actions, it creates a powerful feedback. In that case a stopping mechanism is often used to increase the price of new upgrades to an ability.