Improving with Southbeach

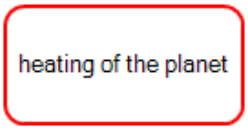
Howard Smith

This is the first of a series of articles that explains how to use Southbeach Notation.

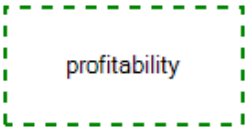
Southbeach was developed to complement other business and engineering diagrams, by adding information that points to needed improvements, e.g. in an enterprise architecture, a business process or a technical system.

Fortunately, Southbeach notation is not difficult to learn. The visual building blocks are easy to remember – green for something you want, red for something you don't want, dashed line for an insufficient quantity, an arrow to signify one factor increasing another, etc.

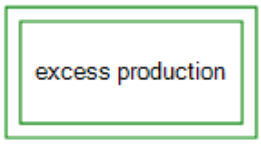
Most problems or challenges, as well as differences of opinion, can be thought about as a situation that needs improvement. Southbeach provides several ways to express the need for change, such as:



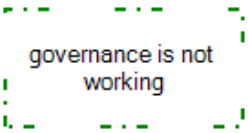
Harmful – you want to eradicate or minimize this factor, find useful side effects, or convert harm into use



Insufficient – you want more of it, or, if harmful, less of it



Surplus – you want to produce less, requiring fewer resources, or, find a use for the excess

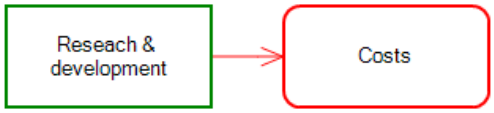


Dysfunctional – it's not working, you need to fix it if it is useful (or perhaps keep it broken if it is considered harmful)

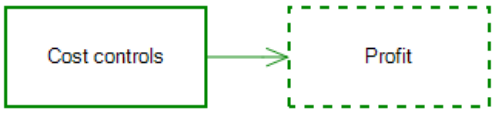
Visual effects can be combined, for example, excessive harm – a red dashed line. The precise interpretation that you put on these primitives is up to you. 'useful' for example could be interpreted as 'contributing' to a goal. The directions suggested for improvement initially look trivial, but they become richer as you add different factors to the system and insert relationships between them.

Southbeach provides a wide range of modeling relationships, where one agent in the system has an

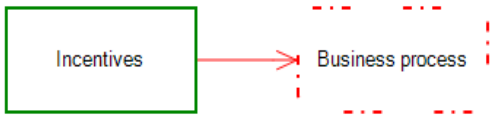
effect on another. Let's look at some examples:



A produces B – increasing A increases B, so if B is harmful, you want to find ways to avoid the need for A in the system, or at least, to keep it low

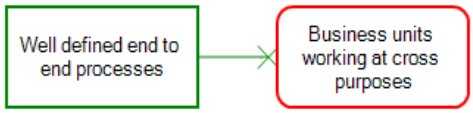


If B is insufficient, but useful, you may want to increase A. But will this work?

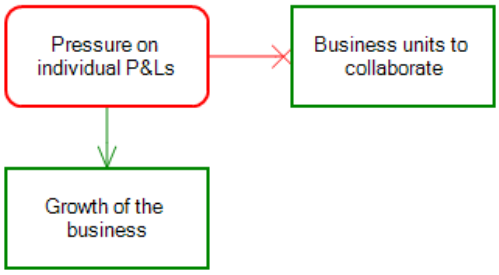


If B is dysfunctional, will increasing A make matters better or worse?

See how combining factors and effects lead to many different improvement strategies. Let's look at another:

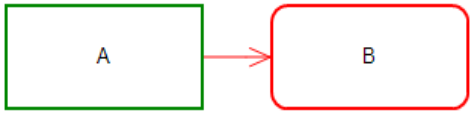


A prevents B – if B is harmful A can prevent its effects in the system – so we certainly don't want A to reduce or disappear – unless we can find other ways to remove or prevent B.

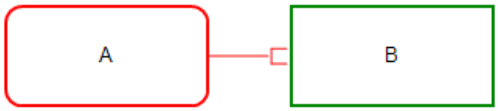


On the other hand, if B is useful, we must target A to find ways to remove or reduce it. Yet we might need that factor for another useful function.

You might have spotted that 'contradictions' are creeping into the model. Resolving a contradiction is an important step in problem solving and innovation. For example:

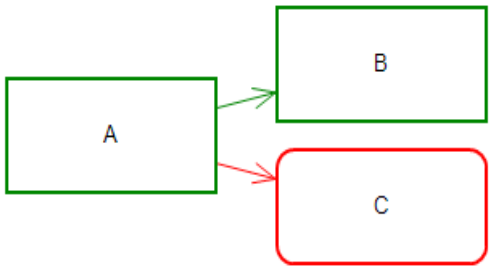


A is useful, but produces harmful B. We want more of A because it is useful, but the more A we have, the more harmful B is generated. What can we do? Do we try to limit B, accepting A will generate it, or do we find an alternative to A in the system?

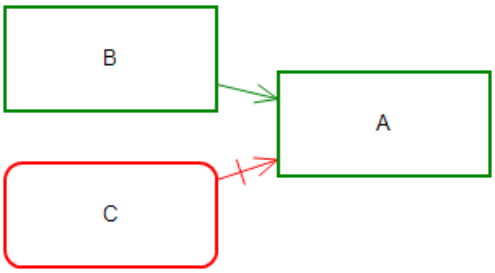


Contradictions can be generated with many other effects. For example, if A is harmful, but contributes to B which is useful. Can we find a substitute for A? Can we find a substitute for B which does not need A?

Contradictions are situations that require us to invent solutions. A colloquial way of saying the same thing is: 'How can we have our cake and eat it?" Contradictions are everywhere:



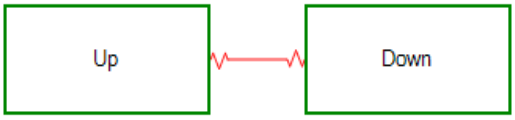
If A produces useful B, but also harmful C, do we substitute it for something else that only produces B, or can we remove it from the system and find a substitute for B.



If B is increasing (producing) A, but C is decreasing (counteracting) A, and A is useful, we need to find ways to break links between these elements, or remove or minimize C.

Note: Even if C were useful, this no doubt has undesirable side effects. Again, a contradiction.

As well as modeling contradictions, Southbeach can model situations in which states of the system are in opposition.



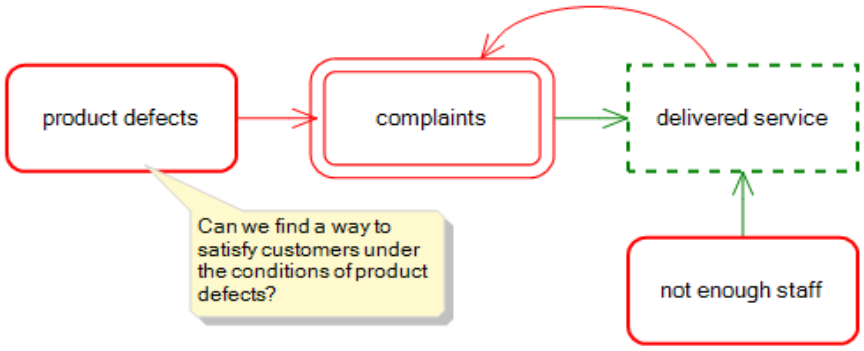
When two useful factors are in opposition, and cannot exist at the same time, a harmful contradiction exists. e.g. a lever cannot be up and down at the same time.

You will no doubt think of many other similar situations.

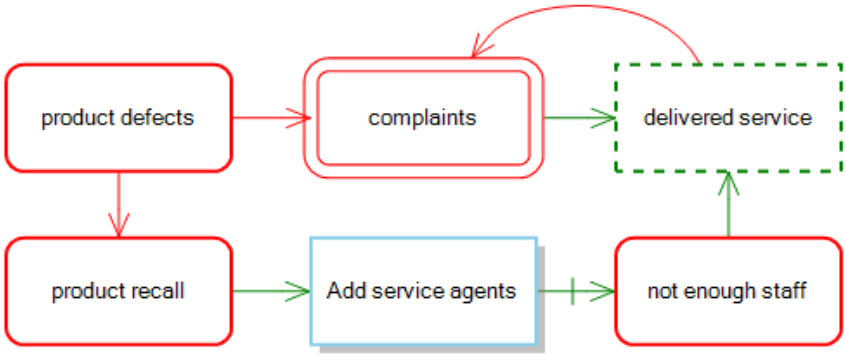
Developing a Southbeach model can be helpful in any area where improvement is sought – process design, business analysis and enterprise architecture. In process design, for example, an activity such as 'delivered service' could be marked as insufficient: partially useful. It could also be marked as harmful if that signifies your 'perspective' on the situation.

The root cause of insufficient service could be modeled by another Southbeach agent, and by doing so yield up a variety of improvement strategies as we have seen earlier in the article. For example, insufficient service arises from too many complaints and not enough staff. The root cause could be identified, e.g. product defects. Similarly, the effect of insufficient service could be modeled: insufficient service causes further customer complaints putting additional load on the system. Thus,

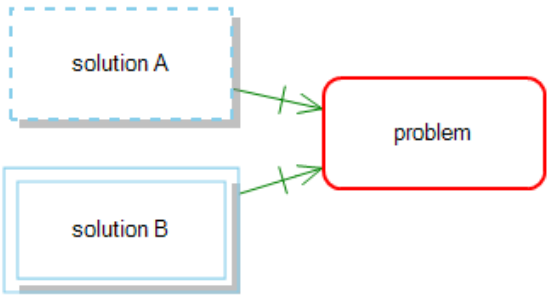
we may be prompted to ask questions such as: can we find a way to satisfy customers under the conditions of product defects?



To illustrate proposals for change, Southbeach supports elements called 'actions'. Think of them as useful proposed solutions added to the system. While useful elements are green, and harmful elements are red, actions are drawn in blue. They allow a designer to donate her proposals for interventions to drive improvements, such as 'adding service agents to cover a period of product recall':



Like other elements, actions can be added to the model using any of the supported effects. For example, it is possible to say A solves harmful element B insufficiently. Thus, it is possible to model partial solutions or over-engineered solutions (surplus).

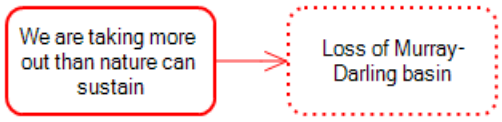


Solution A is insufficient.
Solution B is over engineered.

Do we try to increase the effectiveness of solution A, or find a more efficient version of solution B? Or another solution entirely?

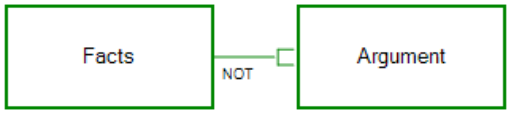
"Everything should be made as simple as possible, but not simpler." – Albert Einstein

Insufficient, surplus and dysfunctional are called 'modifiers'. So far you have seen them applied to agents, but there is no reason why they cannot be applied to effects – for example, excessively trying to counteract a problem – using too many resources. Southbeach also provides additional modifiers – some applying to elements and others only to effects. Here are some examples:

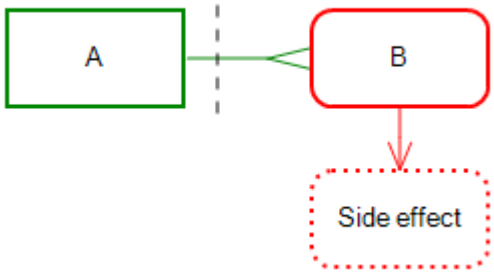


Potential (dotted line) – the element or effect has the potential to exist, but does not (yet) in the current system.

If harmful, we want to avoid the situation. If useful, we want to bring it about: intention.

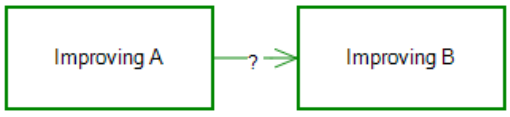


NOT – the effect is not happening, for example, facts do NOT contribute to the argument. Such a statement could prompt unethical directions such as 'How could the facts be changed so as to contribute to the argument?" More correctly: 'What evidence can we seek to support the argument?'

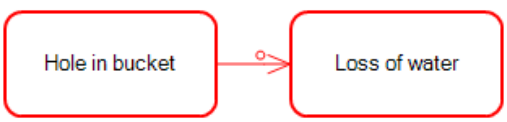


Delayed – the cause-effect relation between two elements contains a damper.

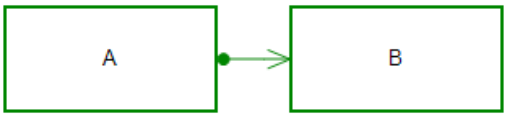
For example, A consumes harmful B, but does not do so fast enough, thus introducing a risk that side effects will occur.



Questionable – the effect may or may not be happening, we are unsure. The model thus prompts questions about evidence or measurement.

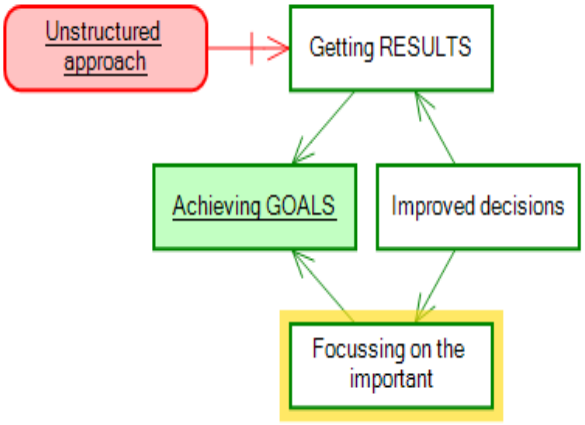


Inevitable – the effect will occur in all circumstances (unless something intervenes to effect it). If the outcome is harmful, it prompts us to look for solutions other than those stated.



Necessary – A is required (mandatory) to produce B – B cannot be produced without A. Thus, Southbeach can express requirements analysis, e.g. Function A is insufficient, but required for Process B.

Southbeach also provides modifiers for denoting goals, risks and focal points within a model:



One or more elements can be marked as a goal, all other elements exist in support of the goal.

A risk is a kind of anti-goal – something we don't want to occur such as system failure or impaired function.

Elements can also be marked with a 'focus' – signifying their special significance in the situation or in problem solving. These elements may not be the goals, but are in some way significant. In the customer service example above, problem analysis could 'focus' on the way complaints are processed.

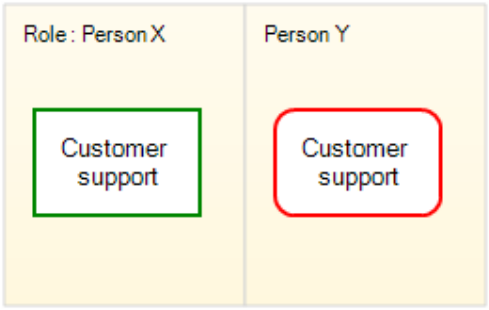
A matter of perspective

What is harmful to you may be useful to someone else. Yin Yang.

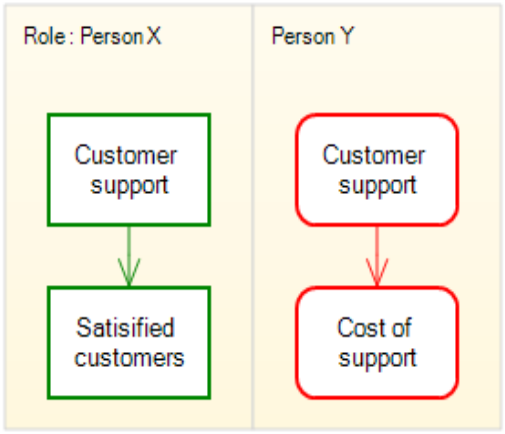
Every useful or harmful element in a Southbeach model may have useful and harmful consequences ad infinitum. Thus, every Southbeach model always represents the perspective of whoever is seeking change or improvement. We know this from our experience in consulting. It is always those who work hard to make change happen that create a future that suits them – not necessarily to the benefit of the whole organization.

To alleviate or solve a complex problem, it is often necessary to develop models from several different perspectives – integrating them later.

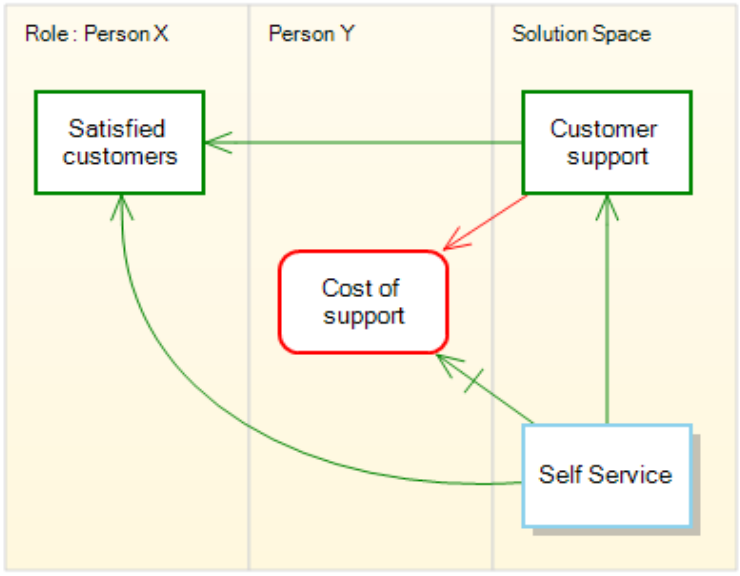
Consider the situation where person X views A as harmful and person Y views A as useful. Here, the model 'separates' into two 'roles':



Can they both be right? Either they are talking about the same A, or different As. If the former, the only interpretation is that they are talking about different elements of A. For example, one person views the customer support function as useful because it produces satisfied customers, another views it as harmful because it represents a cost:



The two perspectives can be reconciled by recognizing that customer support produces both satisfied customers and harmful costs. Customer support is therefore a contradiction: we want satisfied customers but we don't want the cost of serving them. e.g. 'Self service' may be one solution to this contradiction.



In practice it is rarely as easy as this to find an aligned perspective. Even in this case, the language used by X and Y may hide the contradiction they are struggling with. X might speak of satisfied customers and never even mention support, whereas Y might talk about the cost of support in a single phrase. We need to dig under their language, decompose what they say and connect the dots.

Separation

When we decomposed customer support into its useful and harmful elements we created a 'separation'. You are familiar with this from other fields, for example, a programmer decomposes a software package into modules. An enterprise architect decomposes the architecture into the needed services and capabilities. A process designer decomposes a process into activities. Etc.

Southbeach provides several 'dimensions' by which a model can be decomposed:

Separate in time- e.g, was the element useful in the past but is now harmful in the present? Will the element be sufficient or dysfunctional in the future? Trends can be modeled this way. Directions for improvement can be sought in ways that ask questions about why changes occur, suggesting changing the order of events or decomposing the system such that certain operations occur at appropriate times with coordination.

Separate in space – e.g. outside the system, inside the system. Engineering problems may employ this technique, but space can also be understood in conceptual terms, for example, the inner mind, the outer appearance. If an element is useful in one spatial context or harmful in another, spatial transformations are suggested for improving the situation and finding novel solutions.

Separate by structure – e.g. the component is working well (useful) in this part of the system, but malfunctioning in another part of the system. This allows for the modeling of context. Proposals to change the system structure to resolve problems can be considered.

Separate by role – e.g. elements associated with department A, elements associated with department B, corporate elements C.

Separate on probability – e.g. things that are certain to happen, things that could happen, things that can never happen or are unlikely. Combine this with time, and the two dimensional separation can be used to represent 'future horizons' and 'scenarios'. See below.

Separate on (any) conditions – e.g. > 100, <100, hot, cold. Since this can be used to represent any conditions, a wide variety of 'configurations' can be considered in any improvement strategy.

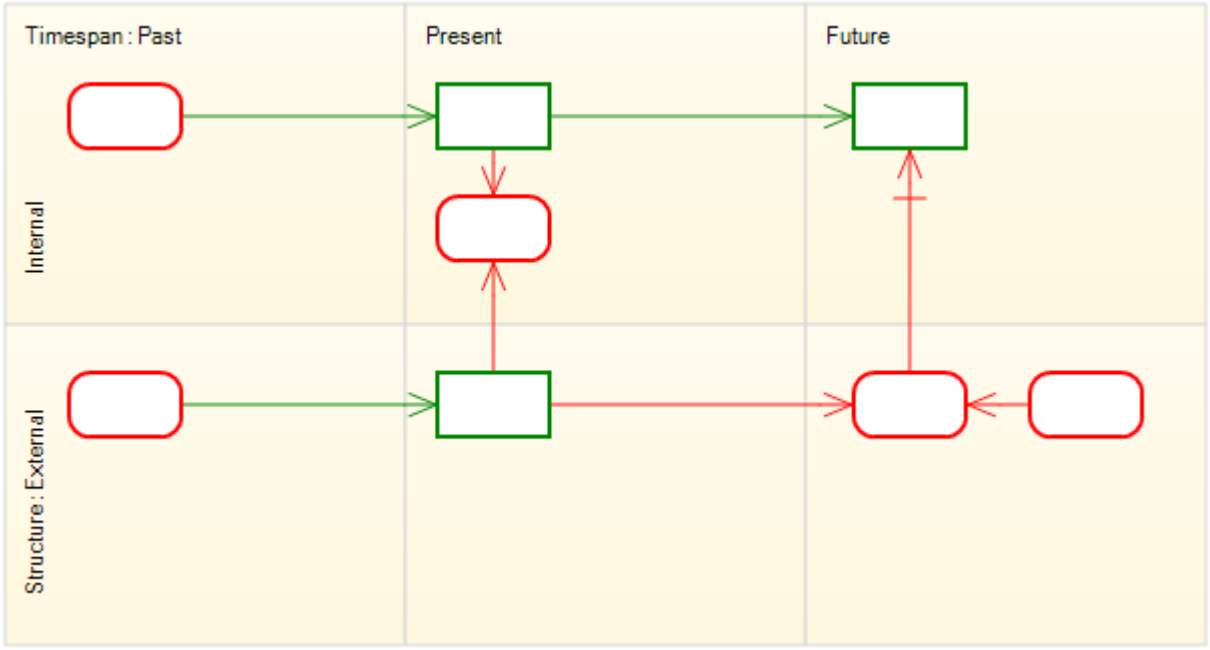
Separate by version – e.g. useful in this product generation, harmful in a future product generation (product or process must evolve). Differences between versions can be modeled to suggest evolutionary improvements.

Separate by perspective – e.g. he views it this way, she views it that way – allowing for the representation of argument. Proposals made by different parties can be modeled and considered.

Separate by aspect – e.g. the product brand is strong, but the product reliability less so. By modeling aspects of the system it is possible to determine which aspects are most important and to develop

solutions for finding those qualities in alternate designs.

Southbeach notation supports one or two dimensions of separation. For example, the past, present and future of the organization (time span) according to external market factors and internal processes (structure of the market).

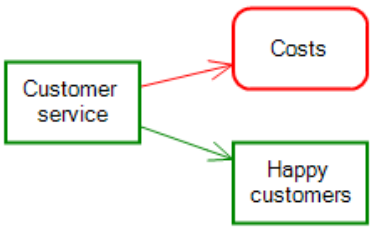


Grids of various kinds divide the modeling canvas into areas representing each zone of separation.

Elements placed on the grid inherit properties. For example, a useful element in the past knows it has been separated from elements placed in the present. This opens up a wide range of creative applications, since the analyst has complete freedom over the values used.

A rules language

Throughout this article we have used examples to show how Southbeach models, based on perspectives, can be used to suggest directions for improvement, problem solving and transformation. Using a rules engine it is possible to automate suggestions of this kind. In a software tool, clicking around the model can generate ideas. For example:



Rules:

produces(useful, useful) “Can we have {to} without {from}?”

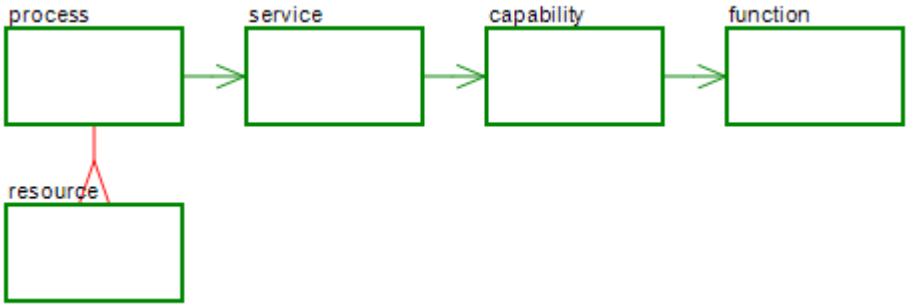
Output:

1. Can we have Happy customers without Customer service?

produces(&A=useful, &B=harmful) produces(&A, &C=useful) "Find a way to get {&C} without the need for {&A} because of the {&B} associated with {&A}"	2. Find a way to get Happy customers without the need for Customer service because of the Costs associated with Customer service
produces(, harmful) "If we cannot remove {from} can {to} be reduced?"	3. If we cannot remove Customer service can Costs be reduced?

A full description of the rules language for Southbeach notation is beyond the scope of this article. The rich notation gives rise to a rich rules language. Rules can refer to agents by name, attributes, modifiers, their placement on grids (i.e. the names of rows or columns) and by their type.

Tags and types



Each element of the system can be given a type – helping to illuminate the operation of the system. Here is one example - a process consuming a resource to provide a service creating a capability to provide a function. (User defined tags will be supported in a future version)

Tool support?

A rich tool that supports Southbeach notation and creative rules – designed for consultants and analysts – is Southbeach Modeller from Southbeach Solutions: www.southbeachinc.com The software is being continuously upgraded, based on user feedback and new applications of the notation.

Southbeach Solutions provides both a free base version of the software, as well as low cost upgrades. The first of these upgrades is MyCreativity, a rules engine that permits a user to automatically generate suggestions and directions for improvement from any model.

About the author

Howard Smith is CTO for CSC Europe (www.csc.com) and author of *Business Process Management: The Third Wave*. His work concerns business process, innovation and problem solving – and their intersection with information technology. Howard created Southbeach with Mark Burnett to fill a need for a perspective-based notation to express change, improvement and intentions and alignment in complex human or technical systems.