

Gamification of Enterprise Strategy

Introduction:

This Hack blends two complementary elements to dramatically advance the state-of-the-art in driving enterprise health and value through business strategy:

1. A comprehensive, facts- and logic-based Universal Strategic Framework™ (USF) to discipline the formulation, validation and execution of complex enterprise strategy; and
2. The addition of Web 2.0 enabled, game play mechanics to bring the USF processes to life and thereby encourage active, real time, broad-gauge stakeholder involvement in fulfilling the enterprise mission and purpose.

Gamified USF methodology will serve *any* transformation well – e.g. post-merger integration of an acquisition or launching a new line of business. However, this Hack focuses on the *extreme* application – i.e. global enterprise strategy to *accelerate purposeful value creation*.

True to Drucker and Goldratt, Compatible with Porter CSV

Enterprise strategy means different things to different people, at different times. USF has no politics or ideology; it merely provides a disciplined framework to aid in the formulation and validation of stepwise actions and results that are individually necessary and collectively sufficient to achieve a goal.

This Hack borrows liberally from both Peter F. Drucker and Eliyahu M. Goldratt, to provide a societal and economic perspective for enterprise strategy, with the expectation that familiarity with and general acceptance of these thought leaders' ideas, will tend to accelerate USF adoption.

Drucker wrote (emphasis added):

*"Business enterprises ... are organs of society. They do not exist for their own sake, but **to fulfill a specific social purpose and to satisfy a specific need of society, community, or individual**. ... There are three tasks, equally important but essentially different, which management has to perform to enable the institution in its charge to function and to make its contribution:*

1. *the **specific purpose and mission of the institution**, whether business enterprise, hospital, or university;*
2. ***making work productive and the worker achieving;***
3. ***managing social impacts and social responsibilities.***

***Business management must always, in every decision and action, put economic performance first.** It can justify its existence and its authority only by the economic results it produces. A business management has failed if it fails to produce economic results. It has failed if it does not supply goods and services desired by the consumer at a price the consumer is willing to pay. It has failed if*

*it does not improve, or at least maintain, the wealth-producing capacity of the economic resources entrusted to it. And this, whatever the economic or political structure or ideology of a society, means **responsibility for profitability.***"

Drucker also wrote: "Every single social and global issue of our day is a business opportunity in disguise"; and "Managers must convert society's needs into opportunities for profitable business".

Insofar as society's needs constitute **mass customer needs**, this Hack encourages enterprises to serve mass customers in meaningful and sustainable ways, which, in all likelihood, constitutes the most profitable long-term application of the USF.

Some may have difficulty with the notion that a complex global enterprise can operate as ONE business system with ONE goal. Nevertheless, Eliyahu M. Goldratt's business bestseller "The Goal" made that case (i.e. that any alternative is suboptimal), rather convincingly, more than three decades ago, using scientific method, based on facts and logic. Goldratt's Theory of Constraints (TOC) has withstood the test and become the lead Operational Excellence discipline for TLS (TOC, Lean Six Sigma). "The Goal" ranks among the **100 Best Business Books of All Time** and one of only eleven titles in the "Management" category.

In agreement with Drucker as to the fundamental necessity of profitability, Goldratt concluded that the ONE Goal should be to "Make more money, now and in the future." In concert with Drucker who advocated Enterprise Value Added (EVA) analysis to get closer to the profit truth, Goldratt also introduced Throughput Accounting (TA; see Appendix A) to eliminate the distortions of Cost Accounting.

Later, with his Viable Vision (see Appendix B) concepts for the "ever-flourishing" enterprise, Goldratt broadened the scope of enterprise performance beyond just "making money", but stuck to the basics of ONE goal. He persuasively argued that every other potentially worthy enterprise goal was either (i) a necessary condition for the ONE goal (and therefore must be addressed as a subordinate objective) or (ii) something enabled by ONE goal attainment (and therefore able to be addressed with the winnings of strategic success).

Recent studies have added empirical backing to Goldratt's logical conclusion favoring ONE goal – i.e. showing that enterprises with fewer (1-3) strategic priorities consistently outperform the rest of the field on revenue growth (see Appendix C).

Finally, Goldratt's concept of the "ever-flourishing" enterprise introduced the Viable Vision notion of "exponential" sales growth, in tandem with stability and security.

Putting these thought leaders' strategic philosophy together, this Hack identifies the ONE goal as: **Accelerate purposeful value creation**, where "purposeful" references clear and compelling enterprise purposes in service of societal needs, thus linking the "purpose motive" and the "profit motive", inextricably.

This Hack's strategic orientation thus parallels the "Creating Shared Value" (CSV) treatise of Michael E. Porter and Mark M. Kramer who wrote:

"We need a more sophisticated form of capitalism, one imbued with a social purpose. But that purpose should arise not out of charity but out of a deeper understanding of competition and economic value creation. The next evolution in the capitalist model recognizes new and better ways to develop products, serve markets and build productive enterprises."

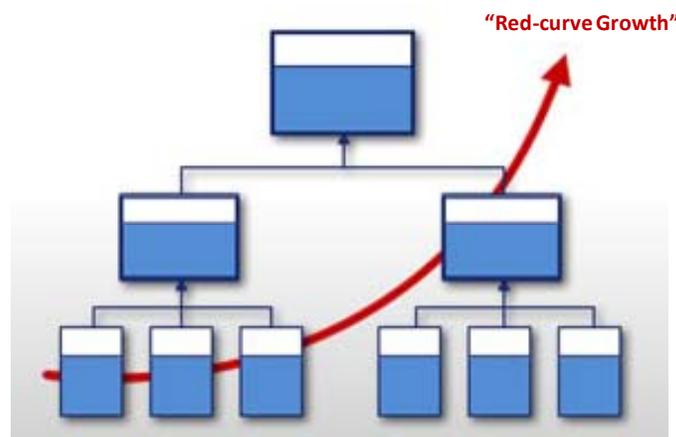
Alignment with Drucker, Goldratt and Porter does not enhance the goal attaining discipline of the Universal Strategic Framework; nor does it add to the fundamental advantages and benefits available from gamification of enterprise strategy. This alignment should, however, underwrite the Hack's implementation, by affording a way to connect with enterprise leaders, most of whom regard Drucker, Porter and Goldratt well and should welcome a rigorous, yet engaging strategic framework that enables their organizations to excel.

Universal Strategic Framework (USF)

The Universal Strategic Framework (USF) serves as a goal-oriented logic tree to discipline, validate, organize and document any enterprise strategic plan, regardless of its complexity.

USF advances the insights of Eli Goldratt's Strategy & Tactic Tree (S&T Tree) insights. Specifically, Goldratt's S&T Tree approach uses necessity and sufficiency logic to break down complex enterprise strategy into all of the logically validated "Steps" that are both individually necessary and collectively sufficient to attain the ultimate goal.

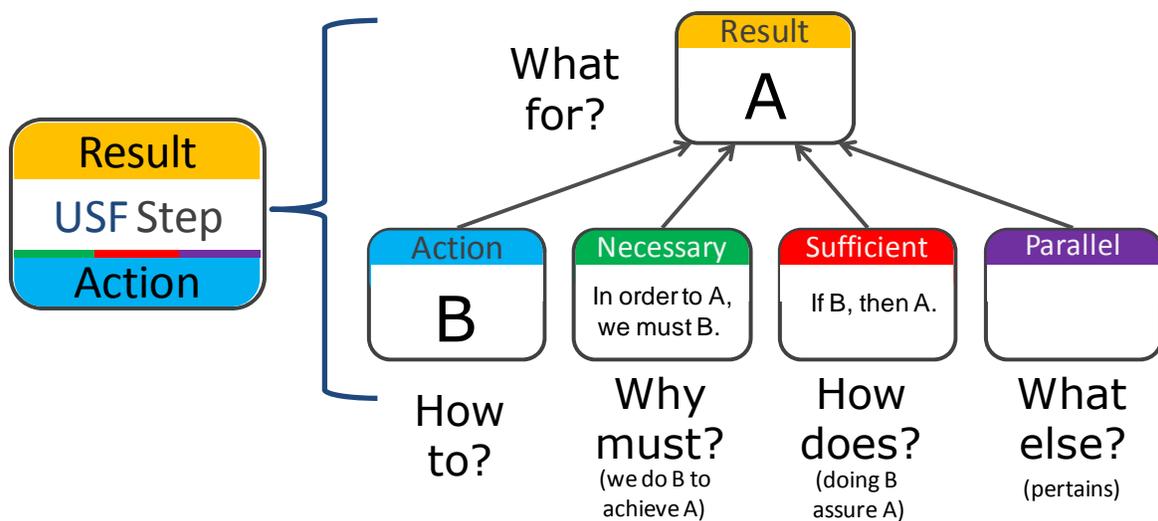
Viable Vision S&T Tree



As previously mentioned, **accelerate purposeful value creation** constitutes the ONE goal for USF enterprise strategy applications. In the context of S&T Trees, Goldratt used the terms "Tactic" and "Strategy" in unconventional and somewhat confusing ways. USF uses Goldratt's one-word definitions of those terms – i.e. "Action" and "Result", respectively, in order avoid confusion arising from legacy meanings (see Appendix D)

Each USF **Step** comprises a single **Action** intended to achieve a single necessary **Result**. Each Step (Action/Result pair) evidences the necessity of the Action to the Result (i.e. "In order to Result, we must Action."), as well as the sufficiency of the Action to the Result (i.e. "If Action, then Result."). Each Step also includes Parallel Assumptions and supplementary information, which facilitate Step implementation and underwrite Step success. *Once complete, each Step constitutes a validated, documented, stand-alone "mini action plan" to achieve a Result that is necessary to ultimate ONE goal attainment.*

USF – Step Composition



USF logic diagrams use necessity and sufficiency logic to connect Steps *upwards* to advance ever higher goal attainment and *downwards* to evidence greater strategic plan detail and extend lines of sight from contributors to the ultimate ONE goal. Asking: "Precisely, what will it take to achieve a Tier 3 Result?", will surface necessary Tier 4 Steps. Asking: "What else; what else?" will ultimately establish the sufficiency of collective Tier 4 Steps. *As practitioners connect USF Steps in this way, the logic tree increasingly becomes a rigorous, robust, in-depth and unambiguous plan of attack for ultimate ONE goal attainment.*

Enterprises may choose any set of Tier 2 USF Steps so long as those Steps prove individually necessary and collectively sufficient to attainment of their ONE goal. Appendix D includes a prototypical USF for enterprise strategy, with the Tier 2 Steps represented as individually necessary and collectively sufficient to *Accelerate purposeful value creation.*

USF broadens the scope of enterprise strategy well beyond the primary focus of Goldratt's S&T Tree, which concentrated on ongoing, TOC-based improvements to the core business.

Like Goldratt, USF acknowledges the importance maintaining a Stable and Secure platform for Enterprise growth, but gives this Result the standing of a Tier 2 Meta – discipline, equal in strategic necessity to the other four USF meta-disciplines: Operational Excellence, Talent Management, Knowledge Creation and Application and Innovation (see Appendix E).

USF broadens the scope of its Operational Excellence meta-discipline beyond TOC, taking advantage of the latest insights regarding the contributions available from TLS (i.e. the complementary application of TOC, Lean and Six Sigma). Moreover USF adds other disciplines, including time management, to the Operational Excellence mix.

Goldratt's S&T Tree templates do not give Talent Management, Thinking/Knowledge and Innovation disciplines the strategic importance that USF accords them as the three meta-disciplines that operate as "performance levers", across three time/growth horizons (Core, Emerging and Promising) to confer value creating advantage on the evolving business system.

USF offers a strategic breakthrough to every business or non-profit. Few companies beyond Goldratt Consulting's modest collection of Viable Vision engagement clients have tapped the advantages available from S&T Tree deployment. USF represents a much more powerful framework (i.e. than S&T Tree), with the inclusion of five meta-disciplines for strategic execution – making USF a unique and powerful addition to the strategic toolkit of any enterprise.

Finally, low-cost, PC-based "Flying Logic" software (see Appendix F) automates all of the various TOC logic diagrams, including S&T Trees, with complete generality. So, Flying Logic can automate the Universal Strategic Framework with its five meta-disciplines. This relieves the enterprise of a considerable administrative in developing and maintaining a large logic tree. Northrop Grumman assisted in the development of Flying Logic, is a co-holder of the copyright and uses the software extensively in its own operations. The availability of Flying Logic software readily enables the gamification of USF, as all the logic diagram requirements have already been programmed.

USF Gamification

The Universal Strategic Framework lends itself to gamification because of (i) the underlying logic that defines game "rules", (ii) the availability of S&T tree facilitation/automation software (Flying Logic) to Web 2.0-enable the game for 24/7/365 access (iii) the raw potential of enterprise strategy to engage and reward players as an online reality game and (iv) the value of using game play mechanics, disciplines and behaviors to underwrite and advance enterprise strategic performance.

Most of the open-source, collaborative, transparent, meritocratic behaviors associated with successful, massively multi-player games fit perfectly with the desired behaviors for perfecting enterprise strategy and execution.

Moreover, every enterprise has tremendous “cognitive surplus” (ref: Clay Shirky) available from its employees because of the “lumpy” (ref: JP Rangaswami) nature of knowledge workers’ workflow. Employees have large reservoirs of tacit knowledge to apply and they universally welcome the opportunity to connect with their enterprise purpose and to see and shape how their individual contributions serve that purpose. Beyond employees, many of the same opportunities and motivations exist with suppliers, customers, shareholders, communities, and society at large.



Forking, crowdsourcing, co-creation, quests, challenges, notifications, badges and lots of other game terminology and mechanics map directly onto the strategic operation of an enterprise with USF.

For example, that part of the logic diagram that conforms to a particular manager’s span of control or sphere of influence could be “forked” by that manager and his or her work group. The USF logic structure facilitates on-the-fly replacement of any Step’s cascading-down Action/Result pairs with a better solution. Responsible managers could issue challenges and quests related to completion times, resource allocations, risk exposures and success likelihoods of satisfying the necessary and sufficient conditions of a Step or collection of interconnected Steps.

Missing pieces of the enterprise’s strategic “puzzle” could be crowdsourced for insights and/or solutions.

Enterprise Learning and Development could be directly aligned with strategic objectives, delivered online and rewarded with badges.

Virtual communities and sub-communities could be formed to correspond to each of the USF’s five Tier 2 Meta-Disciplines as well as each of the three Horizons of Growth. Mashups involving various communities could expose actionable information from diverse internal and external sources.

Naturally, enterprises would need to take measures to secure the enterprise's strategy both for unauthorized access to competitively sensitive enterprise information, to protect employee privacy and to defend cyber attacks.

Employees could adopt game "Personalities" – e.g. ♠Explorers, ♦Achievers, ♥Socializers and ♣Killers – consistent with business initiatives such as Research, Problem Solving, Customer Service and Planned Abandonment, respectively.

Enterprise strategy aligns well with many of the acknowledge game "Boosts", including Engagement, Loyalty, Time Spent, Influence, Fun and User Generated Content. For the USF enterprise strategy game to "Go Viral", might constitute the ultimate boost.

Rewards could include badges or other distinctions that recognized demonstrated expertise, progress and results. Each USF step could have a variety of available rewards associated with formulation, validation and documentation, as well as improvement, advancement and accomplishment.

All the ingredients are there: i.e. the clear need for much better strategy (see Appendix C); a straightforward, rules-based Universal Strategic Framework that can serve any enterprise with extreme effectiveness; a 20th to 21st century paradigm shift that urgently mandates enterprise action; available facilitation software; and multiple motivations to gamify USF to compound enterprise strategic performance.

Appendix A

Throughput Accounting Overview and USF Performance Metrics

Throughput Accounting

From Wikipedia, the free encyclopedia

Throughput Accounting (TA) is a dynamic, integrated, principle-based, and comprehensive management accounting approach that provides managers with decision support information for enterprise optimization. TA is relatively new in management accounting. It is an approach that identifies factors that limit an organization from reaching its goal, and then focuses on simple measures that drive behavior in key areas towards reaching organizational goals. TA was proposed by [Eliyahu M. Goldratt](#)^[1] as an alternative to traditional [cost accounting](#). As such, Throughput Accounting^[2] is neither [cost accounting](#) nor costing because it is cash focused and does not allocate all costs (variable and fixed expenses, including overheads) to products and services sold or provided by an enterprise. Considering the laws of variation, only costs that vary totally with units of output (see definition of T below for TVC) e.g. raw materials, are allocated to products and services which are deducted from sales to determine Throughput. Throughput Accounting is a [management accounting](#) technique used as the performance measures in the [Theory of Constraints](#) (TOC).^[3] It is the business intelligence used for maximizing profits, however, unlike cost accounting that primarily focuses on 'cutting costs' and reducing expenses to make a profit, Throughput Accounting primarily focuses on generating more throughput. Conceptually, Throughput Accounting seeks to increase the velocity or speed at which throughput (see definition of T below) is generated by products and services with respect to an organization's constraint, whether the constraint is internal or external to the organization. Throughput Accounting is the only management accounting methodology that considers constraints as factors limiting the performance of organizations.

Management accounting is an organization's internal set of techniques and methods used to maximize shareholder wealth. Throughput Accounting is thus part of the management accountants' toolkit, ensuring efficiency where it matters as well as the overall effectiveness of the whole organization. It is an internal reporting tool. Outside or external parties to a business depend on accounting reports prepared by financial (public) accountants who apply Generally Accepted Accounting Principles (GAAP) issued by the [Financial Accounting Standards Board](#) (FASB) and enforced by the [U.S. Securities and Exchange Commission](#) (SEC) and other local and international regulatory agencies and bodies.

Throughput Accounting improves profit performance with better management decisions by using measurements that more closely reflect the effect of decisions on three critical monetary variables ([throughput](#), [investment](#) (AKA [inventory](#)), and [operating expense](#) — defined below).

The concepts of Throughput Accounting

Goldratt's alternative begins with the idea that each organization has a goal and that better decisions increase its value. The goal for a profit maximizing firm is easily stated, to increase profit now and in the future. Throughput Accounting applies to not-for-profit organizations too, but they have to develop a goal that makes sense in their individual cases.

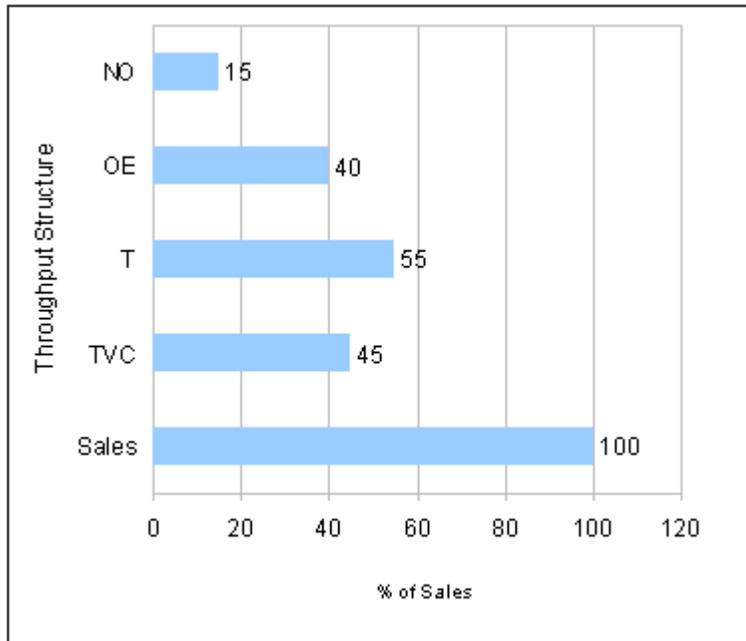
Throughput Accounting also pays particular attention to the concept of 'bottleneck' (referred to as *constraint* in the Theory of Constraints) in the manufacturing or servicing processes.

Throughput Accounting uses three measures of income and expense:

- [Throughput](#) (T) is the rate at which the system produces "goal units." When the goal units are money^[4] (in for-profit businesses), throughput is net sales (S) less totally variable cost (TVC), generally the cost of the raw materials ($T = S - TVC$). Note that T only exists when there is a sale of the product or service. Producing materials that sit in a warehouse does not form part of throughput but rather investment. ("Throughput" is sometimes referred to as "throughput contribution" and has

similarities to the concept of "contribution" in marginal costing which is sales revenues less "variable" costs - "variable" being defined according to the marginal costing philosophy.)

- [Investment](#) (I) is the money tied up in the system. This is money associated with inventory, machinery, buildings, and other assets and liabilities. In earlier [Theory of Constraints](#) (TOC) documentation, the "I" was interchanged between "inventory" and "investment." The preferred term is now only "investment." Note that TOC recommends inventory be valued strictly on totally variable cost associated with creating the inventory, not with additional cost allocations from overhead.
- [Operating expense](#) (OE) is the money the system spends in generating "goal units." For physical products, OE is all expenses except the cost of the raw materials. OE includes maintenance, utilities, rent, taxes and payroll.



The chart illustrates a typical throughput structure of income (sales) and expenses (TVC and OE). $T = \text{Sales} - \text{TVC}$ and $NP = T - \text{OE}$.

Organizations that wish to increase their attainment of [The Goal](#) should therefore require managers to test proposed decisions against three questions. Will the proposed change:

1. Increase throughput? How?
2. Reduce investment ([inventory](#)) (money that cannot be used)? How?
3. Reduce [operating expense](#)? How?

The answers to these questions determine the effect of proposed changes on system wide measurements:

1. [Net profit](#) (NP) = throughput - operating expense = $T - \text{OE}$
2. [Return on investment](#) (ROI) = net profit / investment = NP / I
3. TA [Productivity](#) = throughput / operating expense = T / OE
4. Investment turns (IT) = throughput / investment = T / I

These relationships between financial ratios as illustrated by Goldratt are very similar to a set of relationships defined by [DuPont](#) and [General Motors](#) financial executive [Donaldson Brown](#) about 1920.

Brown did not advocate changes in management accounting methods, but instead used the ratios to evaluate traditional financial accounting data.

Throughput Accounting ^[6] is an important development in modern accounting that allows managers to understand the contribution of constrained resources to the overall profitability of the enterprise. See [cost accounting](#) for practical examples and a detailed description of the evolution of Throughput Accounting.

USF Performance Metrics

Management cannot actually measure value creation in service of the enterprise purpose. Customers buy expected utility when that utility exceeds perceived costs and disutility by an amount favorable to their apparent alternatives, including the “do-nothing” alternative. Customers make their purchase-related value determinations on distinctly individual bases. When an enterprise makes a sale, it cannot tell how much value it created; it can only know that the buyer perceived comparative value and that the enterprise received revenue as a result.

Since enterprises must put economic performance first, they must determine the profitability of aggregated revenues, over time. Throughput Accounting (TA) provides the most timely and most actionable financial information to guide strategic execution. Throughput Accounting is based on Eli Goldratt’s “Theory of Constraints”.

As Thomas Corbett wrote in his book titled: “Throughput Accounting”:

TA “... is **simple** and **logical**; consequently it is understood by all. Not only that, it supplies **trustworthy information fast**, which allows managers to **make good decisions fast**. These are the qualities a management information system should have, and which no other system currently offers.

The ease and speed with which Throughput Accounting provides highly actionable, transparent financial information align well with gamification of enterprise strategy to foster real time, real world performance. Indeed, TA can upgrade financial decision making, throughout the enterprise, with almost immediate benefit and thereafter daily improvement. Moreover, the three straightforward TA measures: Throughput (T), Investment (I) and Operating Expense (OE), readily enable calculations of:

$$\text{Net Profit (NP)} = T - \text{OE}$$

$$\text{Return on Invested Capital (ROIC)} = \text{NP}/I = (T-\text{OE})/I$$

and

$$\text{Productivity (P)} = T/\text{OE}$$

Finally, because Throughput Accounting does not allocate any costs, the method avoids the traps (e.g. Standard Gross Margins and Product Line Profitability) and abuses (e.g. “Inventory Profits”) of conventional Cost Accounting.

Since value creation cannot be measured at the level of an individual sale, it cannot be measured in the aggregate, either. Accordingly, enterprises need a surrogate metric for value creation. This Hack recommends Throughput (T) as that surrogate, based on its service as a single, transparent, actionable performance measure. Indeed, while enterprises can improve financial performance by reducing OE and I, that improvement potential is strictly limited. Throughput, in contrast, has no practical upper bound. Accelerating enterprise Throughput via purposeful value creation serves well as the primary strategic metric. Any enterprise that **accelerates Throughput in pursuit of purposeful value creation** has most likely achieved the goal to: **accelerate purposeful value creation**.

Appendix B

Viable Vision/Strategy & Tactic Tree References

Achieving profit levels which everyone currently believes are unachievable, and achieving it within four years.

Viable Vision

by Eliyahu M. Goldratt

During 2003 I put to the test the reaction of top managers to Viable Vision. But I was careful to expose the reasons for my conviction that this apparently incredible vision is viable. I started by sharing my diagnosis of what is currently blocking the performance of the company. Based on that, using solid cause and effect logic, I deduced the tangible steps that are bound to remove that block. Then I detailed the steps that must be taken in order to capitalize on that breakthrough; the steps that will propel the company to achieve, in less than four years, profit levels which everyone believes are unachievable. Done in this way, the first reaction of top managers was: "This is just common sense, why aren't we doing it?"

Why haven't they done it? How come the prevailing notion is that, unless the company has a unique product or unless the company is very small, it is unrealistic to expect a company to increase its net profit by so much? How come, even though it is possible to construct a Viable Vision for more than half the companies, the prevailing notion is that it is impossible?

The answer is that most people are unaware of the fact that any complex system is based on inherent simplicity. Capitalizing on the inherent simplicity is what enables incredible improvements within a short time.

What is "inherent simplicity?"

To explain this concept we first have to clarify what we refer to as a complex system: "the more data one has to provide in order to fully describe the system, the more complex the system is." If one can fully describe a system in four sentences, it is a simple system. But if one needs a thousand pages to describe it, the system is complex.

How complex is the system you manage? How many pages are needed to describe every process on every part, the relationships with each client, etc? It is no revelation that companies, even small ones, are extremely complex. It is also no revelation that it is difficult to manage a complex system.

So how do we go about managing a complex system? We dissect it into subsystems. Each subsystem is, by definition, less complex than the whole. If you have any hesitation accepting that this is precisely what we do, just look at your organizational chart.

Dissecting a system into subsystems has its price. It leads to miss-synchronization; it leads to harmful local optima and, in some cases, even to the devastating silo mentality. Since our systems are incredibly complex it seems that all that can be done is just to minimize the price; to do the best we can to improve synchronization, and to foster better collaboration between the subsystems.

As long as this is the only option we consider, we'll be under the impression that achieving a significant jump in profit within a relatively short time is unrealistic.

To see the true potential of a company one has to delve deeper into the issue of complexity. What bothers most of us is the fact that part of the data that typifies our system does not

relate to just one component of the system, but to the relationships between two or more components. In other words, the thing that makes our system difficult to manage is that what is done in one place has ramifications in other places; the cause and effect relationships turn our system into almost a maze. But that fact is what provides the key for the solution.

Think about it in the following way. Examine a given system and ask yourself, what is the minimum number of points one has to impact in order to impact the whole system? If the answer is "ten points" then this is a difficult system to manage; it has too many degrees of freedom. It is like attempting to manage a bunch of wild cats. But, if the answer is "just one point" then this system has only one degree of freedom; it is an easy system to manage.

Now, do you agree that the more interdependencies existing between the various components of the system the less degrees of freedom the system has? Considering the enormous complexity of your system it follows that there must be only very few elements that govern the entire system. In other words, the more complex the system is, the more profound is its inherent simplicity.

To capitalize on the inherent simplicity we must be able to identify those few elements that govern the system. Additionally, if we clarify to ourselves the cause and effect relationships between these elements and all other elements of the system, then we can manage the system to achieve a much higher level of performance.

These few elements, the ones dictating the level of performance of the system, are the constraints of the system. This implies that the constraints are also the leverage points of the system. Hence the name I chose for this approach – the Theory Of Constraints - TOC.

Twenty years ago I demonstrated the TOC approach on production systems (manufacturing plants) in my book *The Goal*. Then I demonstrated it on project-based systems in *Critical Chain*. The marketing/strategy of companies is in *Its Not Luck*. If you read any of these books you, most probably, agree that the conclusions are pure common sense, even though they fly in the face of common practice. Moreover, if you are one of the many managers who actually put it into practice you have firsthand experience with the impressive improvements and the surprisingly short time in which you achieved them.

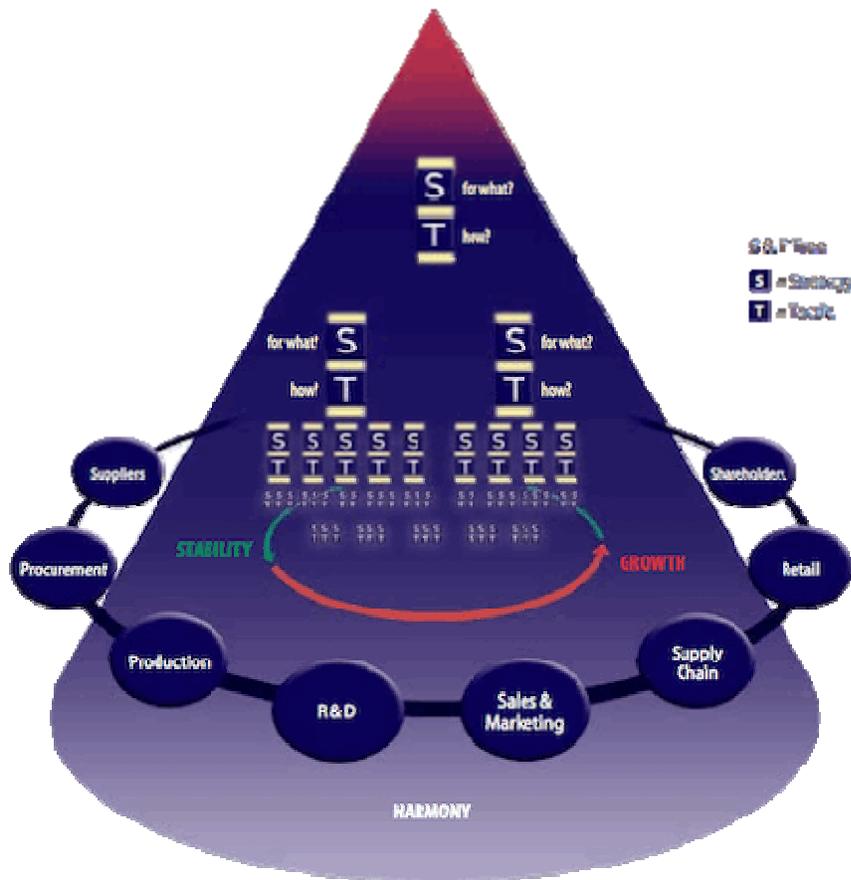
Still, is a Viable Vision possible for your company? Is it feasible to bring your company to achieve, in less than four years, profit levels which everyone believes are unachievable?

The obstacles look insurmountable. For example: it is obvious that such a quantum jump in profitability is impossible without a huge increase in sales. A huge increase in sales can be achieved only if the company will have a new offer that is unrefusable by its markets. Can such a remarkable offer exist? Can the company deliver on such an offer? What investments will be needed? And even if it can be done, is the management team capable of implementing and sustaining such a change?

In these few pages I am unable to answer these questions (and many more). But if you meet with us for a couple of hours I think you will get enough convincing answers to follow my business proposal.

Contact us at info@goldrattconsulting.com to request a meeting with a member of my organization.

Synchronizing all elements to one harmonious composition throughout the organizational hierarchy, across functions, and over time



The TOC Strategy & Tactic Tree (S&T) is the core of a Viable Vision implementation, providing both the blueprint and the roadmap for the company to achieve the Viable Vision objective to become Ever-Flourishing.

An analysis and communication tool which builds a harmonious structure, in which every section of the organization acts for the maximum benefit of the whole.

Constructed to ensure both stability and growth, hand in hand. Choreographs each step in the implementation to yield rapid, tangible results.

Articulates the “what”, the “how” and the frequently elusive “why” for each function and each individual. enables the organization to share ownership of the direction toward the objective.

Appendix C

Recent Enterprise Strategy Perspective – Booz & Co; McKinsey & Company

Stop Chasing Too Many Priorities

8:52 AM Thursday April 14, 2011 by Paul Leinwand and Cesare Mainardi |

If you feel you have too many priorities and claims on your attention, you are hardly alone. A recent survey of 1,800 global executives (see Booz & Company's [Coherence Profiler](#)) that dug into this issue revealed a wide range of related management ailments, including:

Most executives (64%) report they have too many *conflicting* priorities.

The majority of executives (56%) say that allocating resources in a way that really supports the strategy is a significant challenge, especially as companies chase a wide set of growth initiatives. 81% admit that their growth initiatives lead to waste, at least some of the time.

Nearly half (47%) say their company's way of creating value is not well understood by employees or customers.

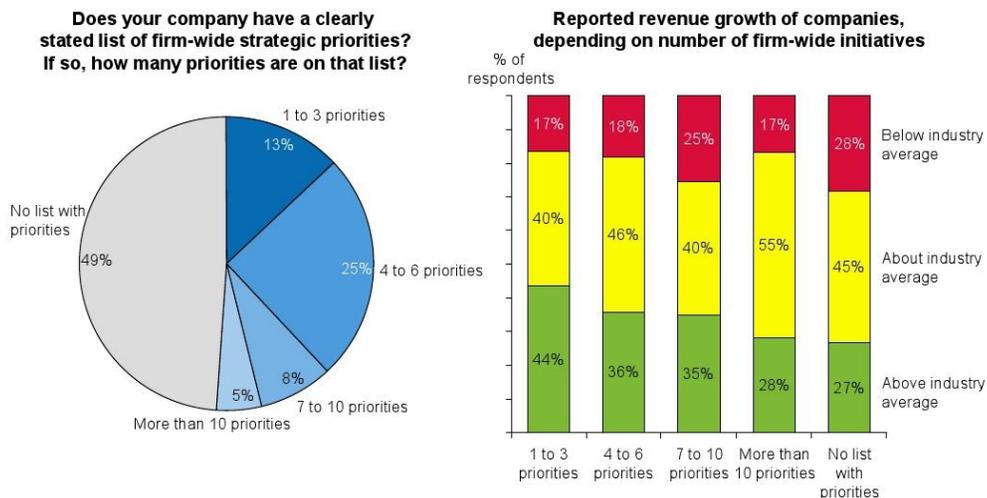
The survey findings suggest that these symptoms stem from companies' *incoherence* — their strong tendency to chase growth initiative after unrelated growth initiative, often with very little success.

The Perils of a Long List of Growth Initiatives

When company leaders develop a new strategy, they usually start by looking for places to grow. This may feel like the right thing to do, but it can be a misleading and even dangerous way to begin a strategic exercise. There are an infinite number of ways that a company can try to grow, and simply brainstorming them will immediately lead to a long list of initiatives. That will soon become an endless litany of priorities, and a large number of conflicting claims on your attention.

Our research reveals, however, that as an executive team's priority list grows, the company's revenue growth in fact declines relative to its peers.

Respondents from firms with fewer firm-wide strategic priorities report higher revenue growth

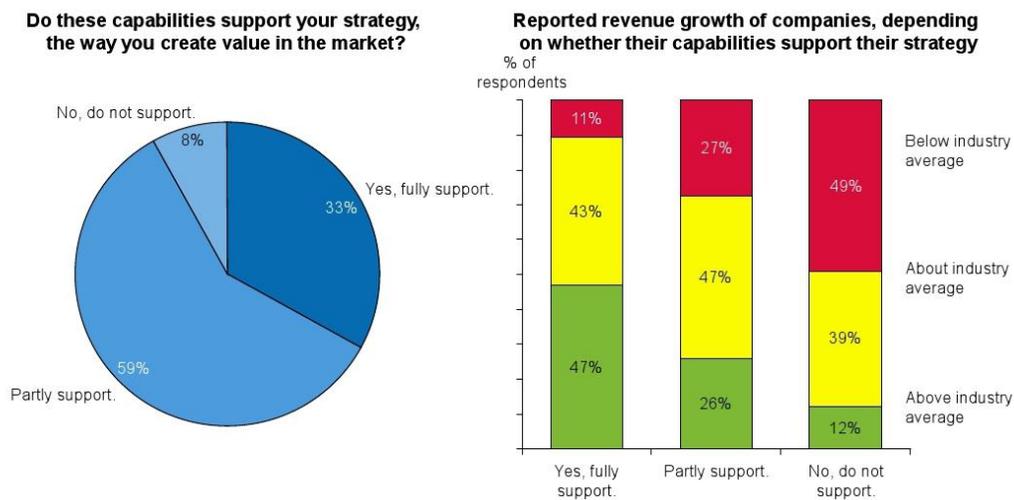


The good news is that the reverse is also true: executives with the most focused set of strategic priorities (one to three priorities) were the most likely to say they had achieved above-average revenue growth.

So the real question executives should be asking is: How can I get focused on the right initiatives for my company?

Another related, and hopeful, finding: About a third of the executives we surveyed say their company's differentiating capabilities "fully support" their strategy. This is a hallmark of what we call "coherence"; it means that all growth initiatives are supported by the same focused investment, effort and attention. These respondents were *three times as likely* to report above-average revenue growth for their companies as the other executives in the survey.

Executives who say their company's capabilities support the strategy are most likely to say revenue growth is above average



Booz & Company
January 10 2011

Sum may not total 100% due to rounding.

So, how do you follow the example of the top-performing companies? Start by asking some basic questions about your own capabilities. What are you great at doing now? If you wanted to truly differentiate yourself from your competitors, what are the three to six most crucial capabilities that you can muster more effectively than everyone else and that would be truly worthy of your attention and resources? The answers can lead to an overarching framework for your strategy that enables better judgment. Only then can you decisively say "yes" or "no" to the vast number of opportunities around you, with the confidence that you are picking initiatives that are not just appealing, but attainable.

We all know instinctively that we cannot do everything - and our companies cannot either. The most pertinent question you can ask is not: "How can I find more business opportunities?" It is: "How can I focus on the opportunities where my company can excel — and then reap the benefits of that discipline?" The key to success is *choosing* the opportunities that are best for *you*, learning to turn down many that seem appealing on the surface — and may even represent huge monetary stakes — but do not offer you a real chance to win.

For more information on developing a capabilities-driven strategy (including several examples and case studies) please refer to our previous posts on: [what it is](#), [why it matters](#), [practical steps](#) to achieving it, and how it can enhance [your legacy as a leader](#).

Paul Leinwand is a Partner in Booz & Company's global consumer, media, and retail practice. He serves as chair of the firm's Knowledge and Marketing Advisory Council. Cesare Mainardi is Managing Director of Booz & Company's North American business and is a member of the firm's Executive Committee. They are co-authors of [The Essential Advantage: How to Win with a Capabilities-Driven Strategy](#), published by Harvard Business Review Press. For more information, visit theessentialadvantage.com.

Making Your Strategy More Relevant

8:01 AM Monday June 20, 2011 by Paul Leinwand and Cesare Mainardi |

Since the idea of a "business strategy" — a long-term plan for growth and profitability — was first developed in the early 1960s, companies around the world have used this tool to pick a competitive position and make their way closer to it.

But many business leaders seem to be losing their confidence in strategy, or at least in their own company's approach to it. This is evident in our ongoing [Booz & Company survey](#), which asks executives from around the world to comment on the results of their strategic initiatives. With more than 2,350 responses so far, the findings suggest a high degree of disillusionment:

- Most of the respondents (53%) don't feel their company's strategy will lead to success.
- Two thirds (67%) say their company's capabilities do not fully support the company's strategy and the way it creates value in the market.
- Only one in five (21%) executives think their company has a "right to win" in all the markets it competes in.

What is going on in these companies? You might say executives are reacting to turbulence: The world is changing so fast that any effort to stick to a strategy will be futile. And in some sense, companies can only profit through speed — adapting immediately to external pressures and moving rapidly to exploit new opportunities.

Yet there are some companies that have prospered for decades while essentially following the same strategy. Among consumer companies, [Alberto Culver](#), whose long-term growth success led Unilever to acquire the company earlier this year, and Coca-Cola come to mind. In financial services, the brokerage [Edward Jones](#) (subscription needed to view article) a good example. These and other success stories suggest that the problem is not with strategy itself as a basis for decision-making.

A more likely explanation is that, in many companies, strategy has grown diffuse over time. Leaders have allowed a host of strategic initiatives to take hold over the years, each developed with the best of intentions. Some strategies were put in place to hold on to an established customer base or to maintain a longstanding profitable business. Others were started in one part of the company as it expanded into new markets. Some may represent the past direction of an acquired business. As they solidified through the years, each of the strategies established a legacy within the company, along with adherents, supporters, and functional investments.

The resulting incoherence is evident in the survey findings. Almost two-thirds of the executives who have responded so far say their biggest frustration is "having too many conflicting priorities." An even greater majority — 82% — say that their growth initiatives lead to waste at least some of the time. Experience suggests that, if anything, these results are understating the problem. For example, how many of the following strategic planning practices have you seen yourself?

1. Running multiple strategy projects whose outcomes contradict or undermine each other;

2. Creating strategies for independent functions like IT or sales, without clearly demonstrating how these relate to the overall company's priorities;
3. Chasing growth as your highest priority, and thus making expensive commitments to new products or projects that turn out to be riskier than expected and that take away focus and investment from the core business;
4. Establishing a strategy based primarily on annual budget decisions, without investing in the capabilities you need to compete;
5. Benchmarking competitors to make strategic investment decisions, ultimately leading to a lack of differentiation (if everyone followed benchmarks, everyone would compete in the same way); or
6. Setting an aspirational "stretch goal" strategy, without changing the company's practices or approach to execution, and thus providing no viable way of getting there.

It's no wonder that so many business leaders don't feel their company's strategy is going to lead to success — and thus end up muddling through with no overall direction.

The Value of Good Strategy

In their race for growth and their continued efforts to cut costs, many leaders forget the true enabler of profitability, value creation, and competitive advantage: a company's distinctive corporate identity. This identity, as defined by what the company does rather than just what it *sells*, has been built up over time; it is grounded in the company's differentiating capabilities (what it does better than anyone else) and its "way to play" (how it provides value for its chosen customers). A company with a distinctive way to play, and the capabilities to match, has a natural advantage in attracting customers, employees, and investors.

Your own strategy must therefore clearly reflect your company's identity. You need to take into account your company as it is today: What do you do particularly well? How do you create value in the markets you currently serve? Your strategy must then look ahead to your overall chosen direction. How do you expect to create value in the future? What changes do you need to make, overall as one enterprise, to get there?

This is not purely a "market-back" or outward looking approach. Nor is it purely internally focused on your core capabilities. It is both. Only when you identify what you are great at (the few most important capabilities that work together in a system that is very difficult for others to copy) and how this greatness matches with market needs do you have a value-creating strategy.

The more disciplined you can be, looking at these critical questions with an eye for your whole company's strategy, the more relevant and robust your strategy will be. Yes, the world is turbulent. And yes, growth will always be important. But responding to market volatility and the need to grow with multiple, unrelated strategy initiatives will leave you where most executives report to be today: chasing too many strategies and lacking the strength required to win in the marketplace. The only reliable way to earn your right to win is to answer the question, "Who are we going to be?" — and define the company by what it does better to deliver value to customers than any other player.

The perils of bad strategy

Richard Rumelt

Bad strategy abounds, says UCLA management professor Richard Rumelt. Senior executives who can spot it stand a much better chance of creating good strategies.

Horatio Nelson had a problem. The British admiral's fleet was outnumbered at Trafalgar by an armada of French and Spanish ships that Napoleon had ordered to disrupt Britain's commerce and prepare for a cross-channel invasion. The prevailing tactics in 1805 were for the two opposing fleets to stay in line, firing broadsides at each other. But Nelson had a strategic insight into how to deal with being outnumbered. He broke the British fleet into two columns and drove them at the Franco-Spanish fleet, hitting its line perpendicularly. The lead British ships took a great risk, but Nelson judged that the less-trained Franco-Spanish gunners would not be able to compensate for the heavy swell that day and that the enemy fleet, with its coherence lost, would be no match for the more experienced British captains and gunners in the ensuing melee. He was proved right: the French and Spanish lost 22 ships, two-thirds of their fleet. The British lost none.¹

Nelson's victory is a classic example of good strategy, which almost always looks this simple and obvious in retrospect. It does not pop out of some strategic-management tool, matrix, triangle, or fill-in-the-blanks scheme. Instead, a talented leader has identified the one or two critical issues in a situation—the pivot points that can multiply the effectiveness of effort—and then focused and concentrated action and resources on them. A good strategy does more than urge us forward

¹ Nelson himself was mortally wounded at Trafalgar, becoming, in death, Britain's greatest naval hero. The battle ensured Britain's naval dominance, which remained secure for a century and a half.



This article is adapted from Richard Rumelt's *Good Strategy/Bad Strategy: The Difference and Why It Matters*, to be published in July 2011 by Crown Publishing.

toward a goal or vision; it honestly acknowledges the challenges we face and provides an approach to overcoming them.

Too many organizational leaders say they have a strategy when they do not. Instead, they espouse what I call “bad strategy.” Bad strategy ignores the power of choice and focus, trying instead to accommodate a multitude of conflicting demands and interests. Like a quarterback whose only advice to his teammates is “let’s win,” bad strategy covers up its failure to guide by embracing the language of broad goals, ambition, vision, and values. Each of these elements is, of course, an important part of human life. But, by themselves, they are not substitutes for the hard work of strategy.

In this article, I try to lay out the attributes of bad strategy and explain why it is so prevalent. Make no mistake: the creeping spread of bad strategy affects us all. Heavy with goals and slogans, governments have become less and less able to solve problems. Corporate boards sign off on strategic plans that are little more than wishful thinking. The US education system is rich with targets and standards but poor at comprehending and countering the sources of underperformance. The only remedy is for us to demand more from those who lead. More than charisma and vision, we must demand good strategy.

The hallmarks of bad strategy

I coined the term bad strategy in 2007 at a Washington, DC, seminar on national-security strategy. My role was to provide a business and corporate-strategy perspective. The participants expected, I think, that my remarks would detail the seriousness and growing competence with which business strategy was created. Using words and slides, I told the group that many businesses did have powerful, effective strategies. But in my personal experiences with corporate practice, I saw a growing profusion of bad strategy.

In the years since that seminar, I have had the opportunity to discuss the bad-strategy concept with a number of senior executives. In the process, I have condensed my list of its key hallmarks to four points: the failure to face the challenge, mistaking goals for strategy, bad strategic objectives, and fluff.

Failure to face the problem

A strategy is a way through a difficulty, an approach to overcoming an obstacle, a response to a challenge. If the challenge is not defined, it

is difficult or impossible to assess the quality of the strategy. And, if you cannot assess that, you cannot reject a bad strategy or improve a good one.

International Harvester learned about this element of bad strategy the hard way. In July 1979, the company's strategic and financial planners produced a thick sheaf of paper titled "Corporate Strategic Plan: International Harvester." It was an amalgam of five separate strategic plans, each created by one of the operating divisions.

The strategic plan did not lack for texture and detail. Looking, for example, within the agricultural-equipment group—International Harvester's core, dating back to the McCormick reaper, which was a foundation of the company—there is information and discussion about each segment. The overall intent was to strengthen the dealer/distributor network and to reduce manufacturing costs. Market share in agricultural equipment was also projected to increase, from 16 percent to 20 percent.

The 'great pushes' during World War I led to the deaths of a generation of European youths. Maybe that's why motivational speakers are not the staple on the European management-lecture circuit that they are in the United States.

That was typical of the overall strategy, which was to increase the company's share in each market, cut costs in each business, and thereby ramp up revenue and profit. A summary graph, showing past and forecast profit, forms an almost perfect hockey stick, with an immediate recovery from decline followed by a steady rise.

The problem with all this was that the plan didn't even mention Harvester's grossly inefficient production facilities, especially in its agricultural-equipment business, or the fact that Harvester had the *worst* labor relations in US industry. As a result, the company's profit margin had been about one-half of its competitors' for a long time. As a corporation, International Harvester's main problem was its inefficient work organization—a problem that would not be solved by investing in new equipment or pressing managers to increase market share.

By cutting administrative overhead, Harvester boosted reported profits for a year or two. But following a disastrous six-month strike, the company quickly began to collapse. It sold off various businesses—including its agricultural-equipment business, to Tenneco. The truck division, renamed Navistar, is today a leading maker of heavy trucks and engines.

To summarize: if you fail to identify and analyze the obstacles, you don't have a strategy. Instead, you have a stretch goal or a budget or a list of things you wish would happen.

Mistaking goals for strategy

A few years ago, a CEO I'll call Chad Logan asked me to work with the management team of his graphic-arts company on "strategic thinking." Logan explained that his overall goal was simple—he called it the "20/20 plan." Revenues were to grow at 20 percent a year, and the profit margin was to be 20 percent or higher.

"This 20/20 plan is a very aggressive financial goal," I said. "What has to happen for it to be realized?" Logan tapped the plan with a blunt forefinger. "The thing I learned as a football player is that winning requires strength and skill, but more than anything it requires the will to win—the drive to succeed. . . . Sure, 20/20 is a stretch, but the secret of success is setting your sights high. We are going to keep pushing until we get there."

I tried again: "Chad, when a company makes the kind of jump in performance your plan envisions, there is usually a key strength you are building on or a change in the industry that opens up new opportunities. Can you clarify what the point of leverage might be here, in your company?"

Logan frowned and pressed his lips together, expressing frustration that I didn't understand him. He pulled a sheet of paper out of his briefcase and ran a finger under the highlighted text. "This is what Jack Welch says," he told me. The text read: "We have found that by reaching for what appears to be the impossible, we often actually do the impossible." (Logan's reading of Welch was, of course, highly selective. Yes, Welch believed in stretch goals. But he also said, "If you don't have a competitive advantage, don't compete.")

The reference to "pushing until we get there" triggered in my mind an association with the great pushes of 1915–17 during World War I, which led to the deaths of a generation of European youths. Maybe

that's why motivational speakers are not the staple on the European management-lecture circuit that they are in the United States. For the slaughtered troops did not suffer from a lack of motivation. They suffered from a lack of competent strategic leadership. A leader may justly ask for "one last push," but the leader's job is more than that. The job of the leader—the strategist—is also to create the conditions that will make the push effective, to have a strategy worthy of the effort called upon.

Bad strategic objectives

Another sign of bad strategy is fuzzy strategic objectives. One form this problem can take is a scrambled mess of things to accomplish—a dog's dinner of goals. A long list of things to do, often mislabeled as strategies or objectives, is not a strategy. It is just a list of things to do. Such lists usually grow out of planning meetings in which a wide variety of stakeholders suggest things they would like to see accomplished. Rather than focus on a few important items, the group sweeps the whole day's collection into the strategic plan. Then, in recognition that it is a dog's dinner, the label "long term" is added, implying that none of these things need be done today. As a vivid example, I recently had the chance to discuss strategy with the mayor of a small city in the Pacific Northwest. His planning committee's strategic plan contained 47 strategies and 178 action items. Action item number 122 was "create a strategic plan."

A second type of weak strategic objective is one that is "blue sky"—typically a simple restatement of the desired state of affairs or of the challenge. It skips over the annoying fact that no one has a clue as to how to get there. A leader may successfully identify the key challenge and propose an overall approach to dealing with the challenge. But if the consequent strategic objectives are just as difficult to meet as the original challenge, the strategy has added little value.

Good strategy, in contrast, works by focusing energy and resources on one, or a very few, pivotal objectives whose accomplishment will lead to a cascade of favorable outcomes. It also builds a bridge between the critical challenge at the heart of the strategy and action—between desire and immediate objectives that lie within grasp. Thus, the objectives that a good strategy sets stand a good chance of being accomplished, given existing resources and competencies.

Fluff

A final hallmark of mediocrity and bad strategy is superficial abstraction—a flurry of fluff—designed to mask the absence of thought.

Fluff is a restatement of the obvious, combined with a generous sprinkling of buzzwords that masquerade as expertise. Here is a quote from a major retail bank's internal strategy memoranda: "Our fundamental strategy is one of customer-centric intermediation." Intermediation means that the company accepts deposits and then lends out the money. In other words, it is a bank. The buzzphrase "customer-centric" could mean that the bank competes by offering better terms and service, but an examination of its policies does not reveal any distinction in this regard. The phrase "customer-centric intermediation" is pure fluff. Remove the fluff and you learn that the bank's fundamental strategy is being a bank.

Why so much bad strategy?

Bad strategy has many roots, but I'll focus on two here: the inability to choose and template-style planning—filling in the blanks with "vision, mission, values, strategies."

The inability to choose

Strategy involves focus and, therefore, choice. And choice means setting aside some goals in favor of others. When this hard work is not done, weak strategy is the result. In 1992, I sat in on a strategy discussion among senior executives at Digital Equipment Corporation (DEC). A leader of the minicomputer revolution of the 1960s and 1970s, DEC had been losing ground for several years to the newer 32-bit personal computers. There were serious doubts that the company could survive for long without dramatic changes.

To simplify matters, I will pretend that only three executives were present. "Alec" argued that DEC had always been a computer company and should continue integrating hardware and software into usable systems. "Beverly" felt that the only distinctive resource DEC had to build on was its customer relationships. Hence, she derided Alec's "Boxes" strategy and argued in favor of a "Solutions" strategy that solved customer problems. "Craig" held that the heart of the computer industry was semiconductor technology and that the company should focus its resources on designing and building better "Chips."

Choice was necessary: both the Chips and Solutions strategies represented dramatic transformations of the firm, and each would require wholly new skills and work practices. One wouldn't choose either risky alternative unless the status quo Boxes strategy was likely to fail. And one wouldn't choose to do both Chips and Solutions at the same



Scan through template-style planning documents and you will find pious statements of the obvious presented as if they were decisive insights.

time, because there was little common ground between them. It is not feasible to do two separate, deep transformations of a company's core at once.

With equally powerful executives arguing for each of the three conflicting strategies, the meeting was intense. DEC's chief executive, Ken Olsen, had made the mistake of asking the group to reach a consensus. It was unable to do that, because a majority preferred Solutions to Boxes, a majority preferred Boxes to Chips, and a majority also preferred Chips to Solutions. No matter which of the three paths was chosen, a majority preferred something else. This dilemma wasn't unique to the stand-off at DEC. The French philosopher Nicolas de Condorcet achieved immortality by first pointing out the possibility of such a paradox arising, and economist Kenneth Arrow won a Nobel Prize for showing that "Condorcet's paradox" cannot be resolved through cleverer voting schemes.

Not surprisingly, the group compromised on a statement: "DEC is committed to providing high-quality products and services and being a leader in data processing." This fluffy, amorphous statement was, of course, not a strategy. It was a political outcome reached by individuals who, forced to reach a consensus, could not agree on which interests and concepts to forego.

Ken Olsen was replaced, in June 1992, by Robert Palmer, who had headed the company's semiconductor engineering. Palmer made it clear that the strategy would be Chips. One point of view had finally won. But by then it was five years too late. Palmer stopped the losses for a while but could not stem the tide of ever more powerful personal computers that were overtaking the firm. In 1998, DEC was acquired by Compaq, which, in turn, was acquired by Hewlett-Packard three years later.

Template-style strategy

The Jack Welch quote about “reaching for what appears to be the impossible” is fairly standard motivational fare, available from literally hundreds of motivational speakers, books, calendars, memo pads, and Web sites. This fascination with positive thinking has helped inspire ideas about charismatic leadership and the power of a shared vision, reducing them to something of a formula. The general outline goes like this: the transformational leader (1) develops or has a vision, (2) inspires people to sacrifice (change) for the good of the organization, and (3) empowers people to accomplish the vision.

By the early 2000s, the juxtaposition of vision-led leadership and strategy work had produced a template-style system of strategic planning. (Type “vision mission strategy” into a search engine and you’ll find thousands of examples of this kind of template for sale and in use.) The template looks like this:

The Vision. Fill in your vision of what the school/business/nation will be like in the future. Currently popular visions are to be the best or the leading or the best known.

The Mission. Fill in a high-sounding, politically correct statement of the purpose of the school/business/nation. Innovation, human progress, and sustainable solutions are popular elements of a mission statement.

The Values. Fill in a statement that describes the company’s values. Make sure they are noncontroversial. Key words include “integrity,” “respect,” and “excellence.”

The Strategies. Fill in some aspirations/goals but call them strategies. For example, “to invest in a portfolio of performance businesses that create value for our shareholders and growth for our customers.”

This template-style planning has been enthusiastically adopted by corporations, school boards, university presidents, and government agencies. Scan through such documents and you will find pious statements of the obvious presented as if they were decisive insights. The enormous problem all this creates is that someone who actually wishes to conceive and implement an effective strategy is surrounded by empty rhetoric and bad examples.

The kernel of good strategy

By now, I hope you are fully awake to the dramatic differences between good and bad strategy. Let me close by trying to give you a leg up in crafting good strategies, which have a basic underlying structure:

1. A diagnosis: an explanation of the nature of the challenge. A good diagnosis simplifies the often overwhelming complexity of reality by identifying certain aspects of the situation as being the critical ones.

2. A guiding policy: an overall approach chosen to cope with or overcome the obstacles identified in the diagnosis.

3. Coherent actions: steps that are coordinated with one another to support the accomplishment of the guiding policy.

I'll illustrate by describing Nvidia's journey from troubled start-up to market leader for 3-D graphics chips. Nvidia's first product, a PC add-in board for video, audio, and 3-D graphics, was a commercial failure. In 1995, rival start-up 3Dfx Interactive took the lead in serving the burgeoning demand of gamers for fast 3-D graphics chips. Furthermore, there were rumors that industry giant Intel was thinking about introducing its own 3-D graphics chip. The diagnosis: "We are losing the performance race."

Nvidia CEO Jen-Hsun Huang's key insight was that given the rapid state of advance in 3-D graphics, releasing a new chip every 6 months, instead of at the industry standard rate of every 18 months, would make a critical difference. The guiding policy, in short, was to "release a faster, better chip three times faster than the industry norm."

To accomplish this fast release cycle, the company emphasized several coherent actions: it formed three development teams, which worked on overlapping schedules; it invested in massive simulation and emulation facilities to avoid delays in the fabrication of chips and in the development of software drivers; and, over time, it regained control of driver development from the branded add-in board makers.

Over the next decade, the strategy worked brilliantly. Intel introduced its 3-D graphics chip in 1998 but did not keep up the pace, exiting the business of discrete 3-D graphics chips a year later. In 2000, cred-

itors of 3Dfx initiated bankruptcy proceedings against the company, which was struggling to keep up with Nvidia. In 2007, *Forbes* named Nvidia the “Company of the Year.”²



Despite the roar of voices equating strategy with ambition, leadership, vision, or planning, strategy is none of these. Rather, it is coherent action backed by an argument. And the core of the strategist’s work is always the same: discover the crucial factors in a situation and design a way to coordinate and focus actions to deal with them. ○

²The effectiveness of even good strategies isn’t permanently assured. ATI, now part of AMD, has become a powerful competitor in graphics processing units, and Nvidia has been challenged in the fast-growing mobile-graphics business, where cost is often more important than performance.

Richard Rumelt is the Harry and Elsa Kunin Professor of Business and Society at the UCLA Anderson School of Management.



Appendix D

Universal Strategic Framework (USF) Highlights

Universal Strategic Framework (USF)

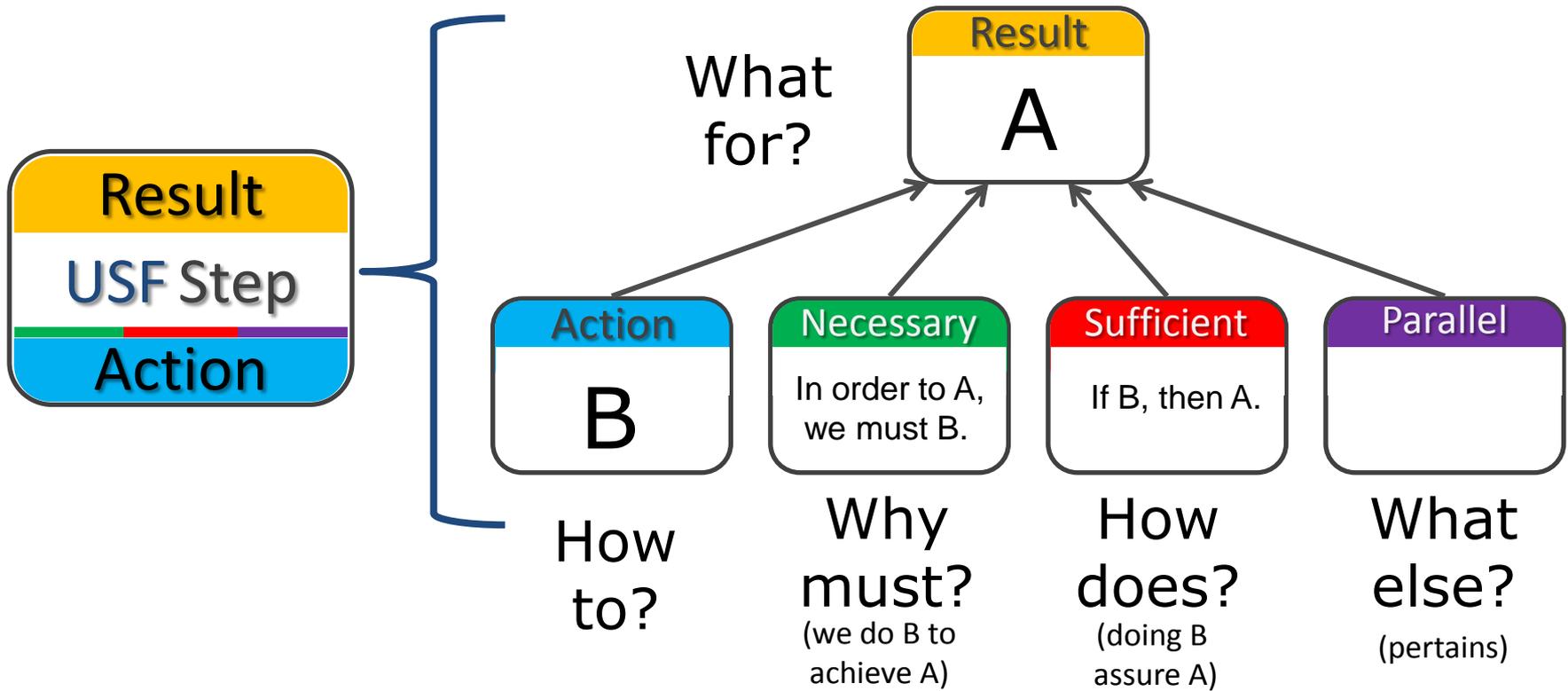
Review of Highlights

USF – Step Composition



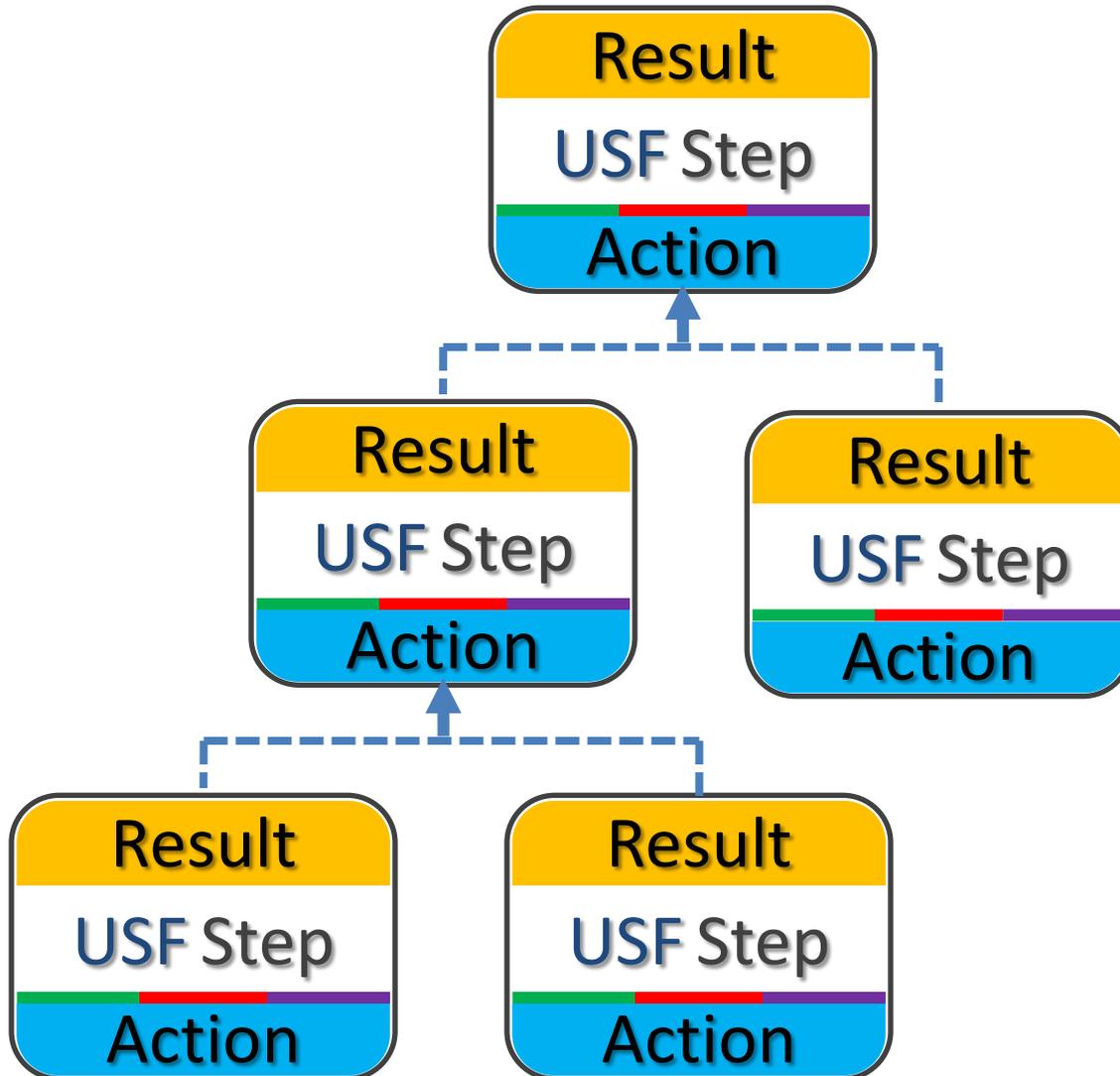
A **USF Step** pairs a single **Result** with a single **Action** which is both **necessary** and **sufficient** to achieve that **Result**. Visually, the green band depicts the necessity condition and the red band depicts the sufficiency condition. The purple band represents the parallel assumption(s) that underwrite success – i.e. **Action** achieves intended **Result**.

USF – Step Composition



When complete, each **USF Step** stands on its own, as a fully validated means to achieve a particular **Result**, with complete transparency – i.e. all of the associated facts and logic are fully available for inspection.

USF – Step Connections

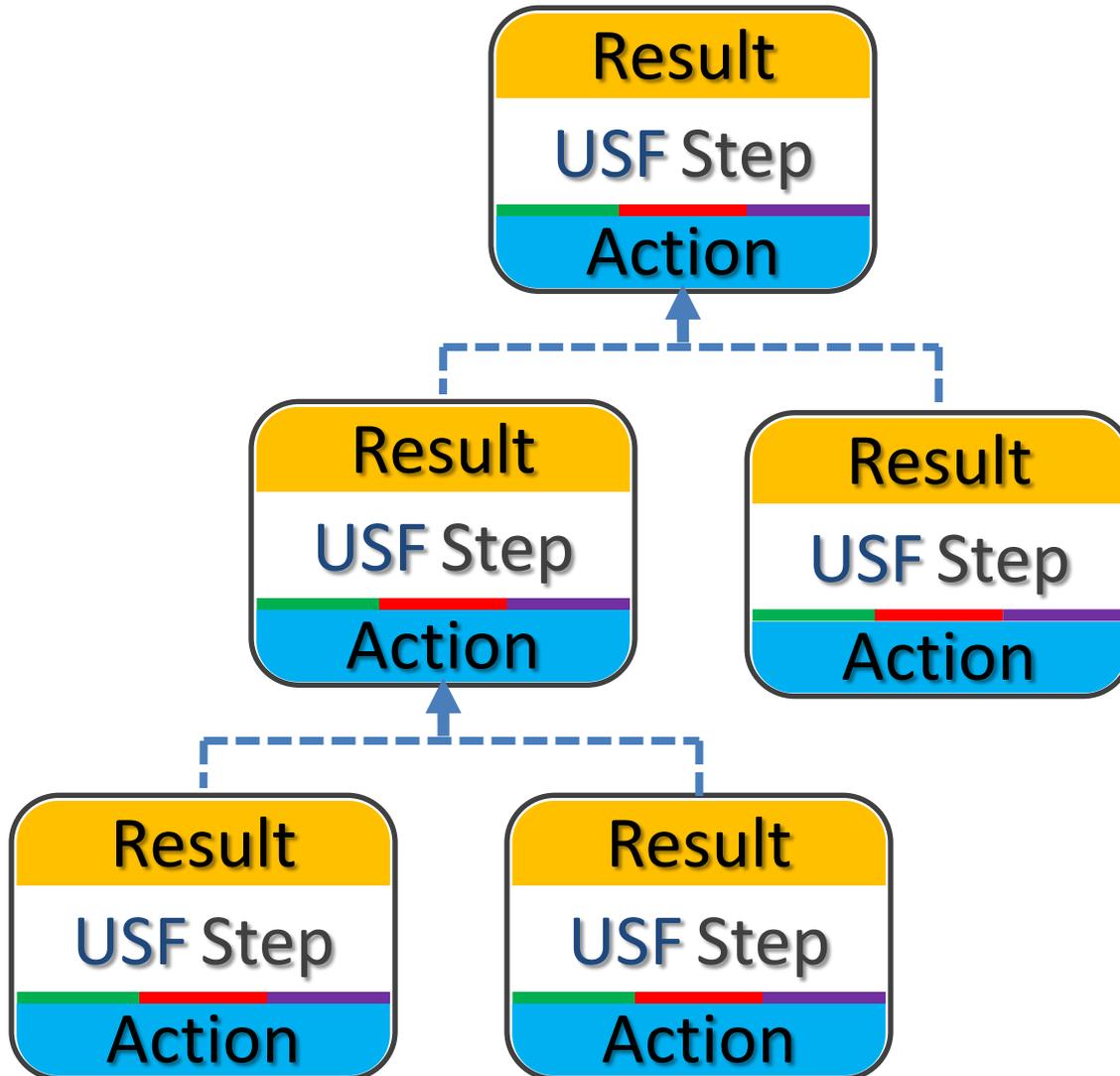


A USF dashed line indicates the necessity of lower tier Steps to the next higher tier Step. A solid arrow shows the collective sufficiency of the lower Steps to the next higher tier Step.

There is no limit on the number of necessary Steps which may be needed to satisfy the sufficiency condition.

A USF logic diagram always connects all of the Steps of any strategy in exactly this way.

USF – Step Connections

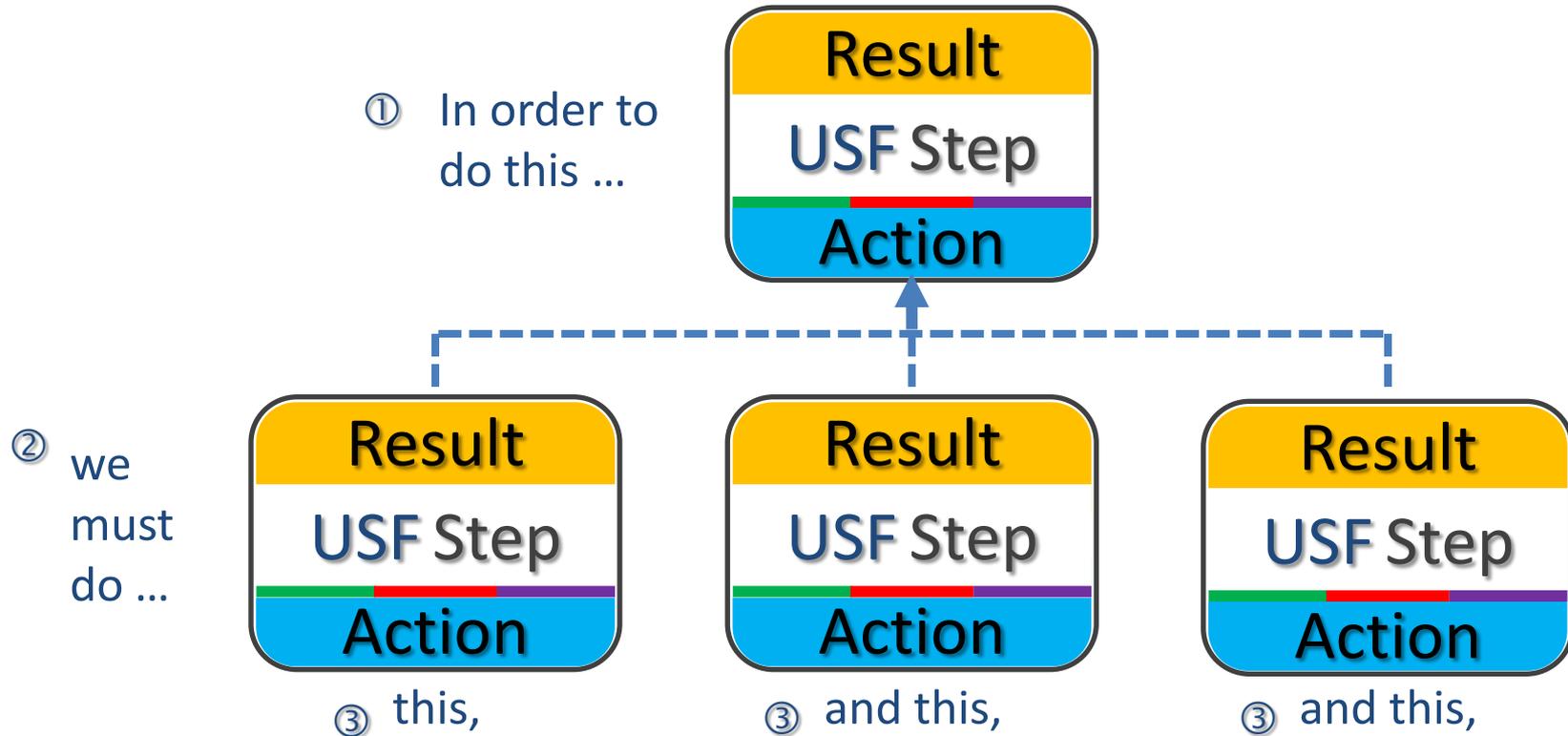


USF Step connections flow upward toward the ultimate enterprise Result (goal or strategic objective).

Step connections also cascade downward into increasing levels of detail that communicate the enterprise strategy and attest to its validity.

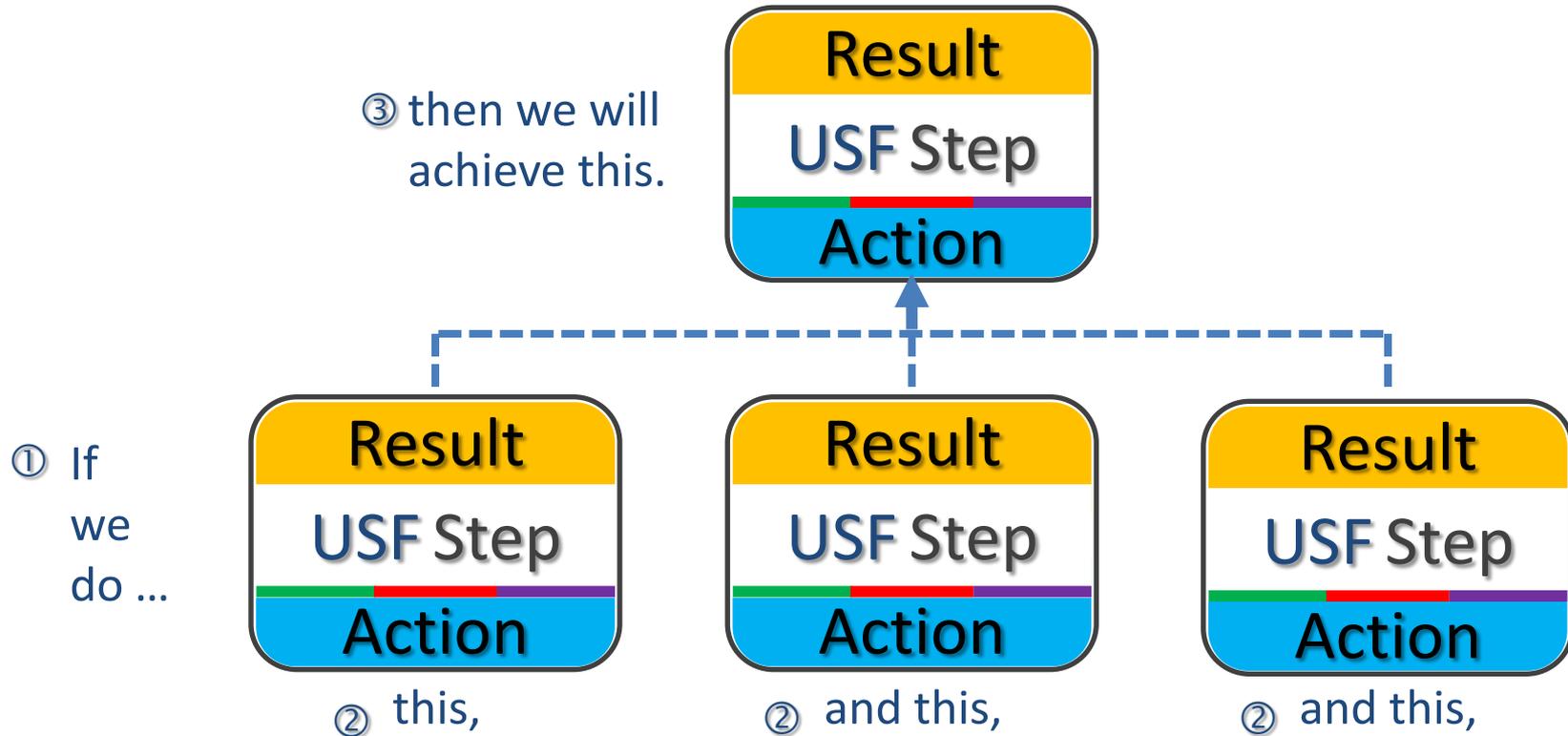
The connection paths clearly show how lower tier Steps contribute to achieving the enterprise goal (i.e. line of sight).

USF – Step Connections



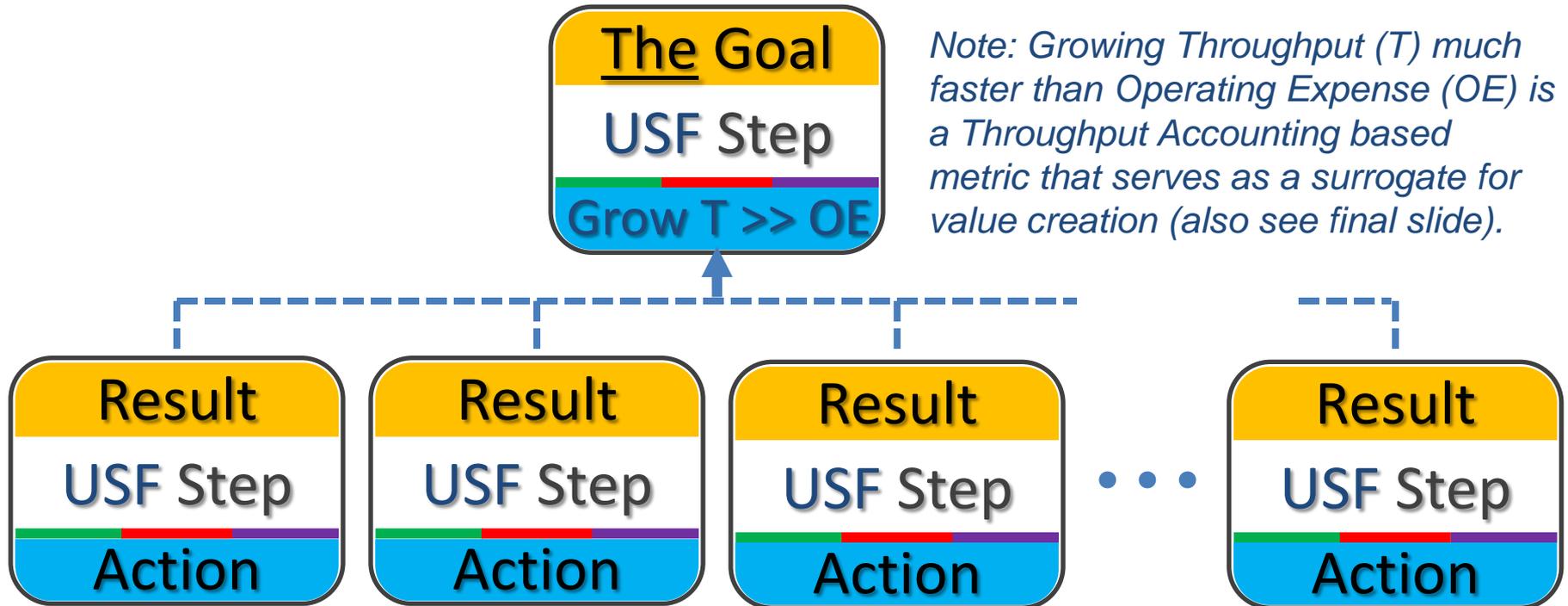
USF logic diagrams make it easy to verbalize **necessity** conditions.

USF – Step Connections



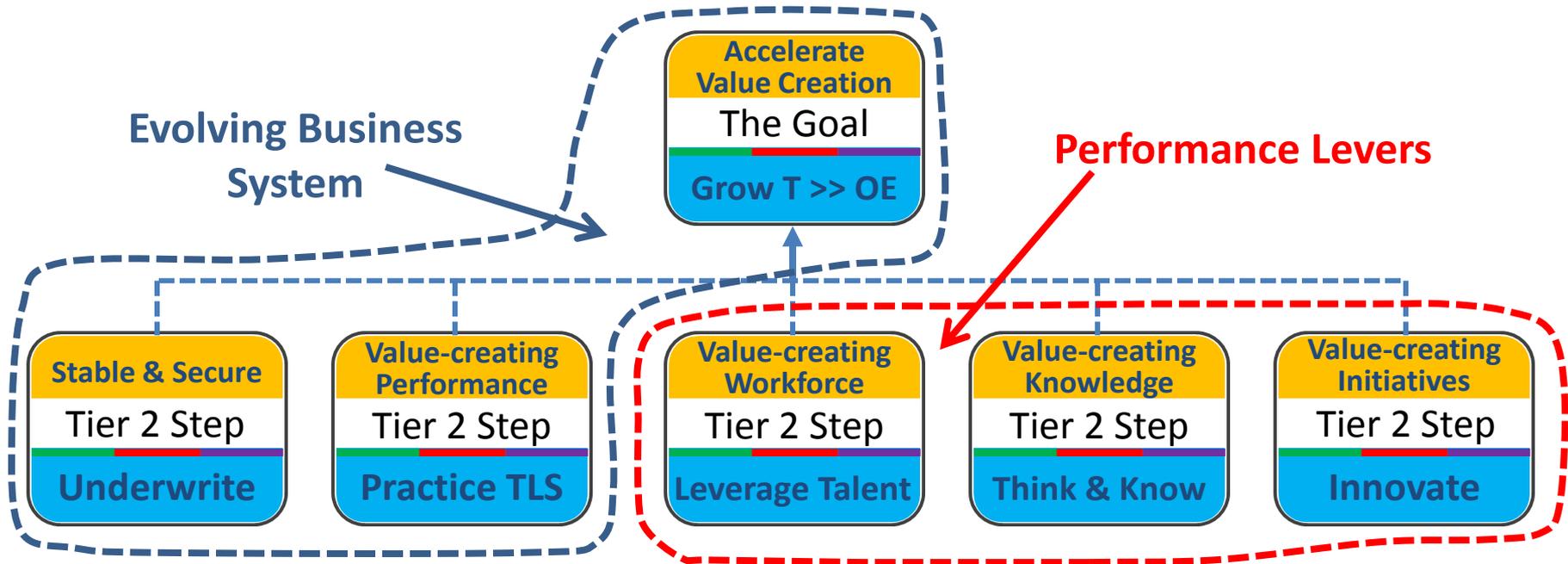
USF logic diagrams make it easy to verbalize **sufficiency** conditions.

USF – One System, One Goal



USF logic diagrams focus attentions, resources and efforts on The Goal of every business: **Accelerate Purposeful Value Creation** – i.e. “Create more value, now and in the future, in service of the enterprise purpose”.

Universal Strategic Framework (USF)



Most businesses should share the same goal: **Accelerate Purposeful Value Creation**. They should also share generic means for achieving that goal, in the form of a stable, secure, proficient and evolving business system, as well as performance levers capable of systematically conferring value-creating advantage on their business system.

The evolving business system should operate in (and on) the present. The performance levers should operate across three distinct business development time/growth horizons: Core, Emerging and Promising.

Universal Strategic Framework (USF)

One Business System



One Enterprise Goal

Customers buy utility and they perceive comparative value on distinctly individual terms. That makes customer-centric value hard to gauge, with precision.

Only through the aggregation of customer purchases, over time, can enterprises discern trends in customer-centric value delivery. Yet, while aggregation may reveal trends, it also tends to blur distinctions between customers and their underlying choices.

Throughput (T) provides a precisely measurable surrogate for value creation. Moreover, focusing on Throughput, while managing Operating Expense (OE) and Investment (I), helps firms make sound business decisions that respect vital interests in addition to delighting customers.

Appendix E

Universal Strategic Framework (USF) – Five Tier 2 Meta-disciplines

Five Tier 2 USF Meta-disciplines

The Universal Strategic Framework applies five meta-disciplines, in concert, to underwrite strategic, ONE Goal attainment in any enterprise, in any industry.

The term meta-discipline acknowledges the inclusion of, and prospective subsequent admission of, a number of business disciplines related to the named meta-disciplines, as further explained, below.

Stable and Secure Platform for Business Growth – this meta-discipline addresses the fundamentals of enterprise *capabilities, resources, protections, directions and connections* that underwrite the ability of any organization to demonstrate and sustain competence.

Operational Excellence – Value-creating service of enterprise purpose demands that the organization do well what matters most in the context of its purpose and continuously improve. The Universal Strategic Framework adopts the disciplines of TLS for operational excellence. TLS in itself is a meta-discipline comprising Theory of Constraints, Lean and Six Sigma (Reading list: “The Goal”, “The Ultimate Improvement Cycle”, “Velocity”, “Throughput Accounting” and “The Logical Thinking Process”, among others.) The USF also incorporates a variety of “time and information management” practices into its Operational Excellence meta-discipline

Talent Management – Ultimately people and organization deliver results. Enterprises can dramatically improve purposeful value creation through Talent Management practices directed at everything from *Job Matching, Motivation and Team Building to Managerial Relationships, Leadership Authenticity and Employee Engagement*. (Reading List: “Managing Oneself”, “What It Means to Work Here”, “Tapping the Unrealized Performance Potential of Employee Engagement” “Drive”, “Flow”, among many others).

Thinking-enabled Enterprise™ – The source of wealth is *Knowledge* and employee *Thinking* ability constitutes the ultimate business resource. Successful enterprises can think more, think better and think differently. Thinking ability can be taught, learned and practiced to proficiency. That thinking ability benefits the enterprise but belongs to the individual sets up a wonderful win/win learning and development opportunity. USF Thinking Disciplines include: Lateral Thinking, Parallel Thinking, Design Thinking and TLTP. The Thinking-enabled Enterprise concentrates on the acquisition, application and creation of knowledge that confers advantage in purposeful value creation (Reading List: “Lateral Thinking”, “Six Thinking Hats”, “Change by Design” and “The Logical Thinking Process”, among others)

Innovation – USF regards “Value Innovation” (“Blue Ocean Strategy”) as the lead innovation discipline. The straightforward recipe for success: *deliver unprecedented utility to a mass of buyers at an accessible price and with a profitable business model* will expand the fortunes of any business in any industry at any time. All the other innovation disciplines (e.g. disruptive innovation) are entirely compatible with the VI recipe for success. (Reading List: “Blue Ocean Strategy”, “Innovation and Entrepreneurship”, “10 Rules for Strategic Innovators”, “Innovators Solution”, “Disrupting Class”, among many, many others.)

Vision21's Five Meta-discipline Perspective

To accelerate value creation means that created value will grow at an ever increasing rate (i.e. much like Goldratt's "exponential" sales growth under stable conditions as the Viable Vision conditions for and "Ever-Flourishing" business).

Vision21 regards these five meta-disciplines as *individually necessary and collectively sufficient* to the attainment of the ONE USF goal: *accelerate purposeful value creation*.

All five meta-disciplines must operate across Three Horizons of Growth which McKinsey & Company identified as *Core, Emerging and Promising*. These Horizons have a natural, but not essential time dimension or alignment. The nature of innovation can bring something "Promising" to the fore in a manner that leapfrogs "Emerging", for example.

Vision21's perspective and recommendations notwithstanding, an enterprise can conceivably identify Tier 2 Steps in different numbers and with different Action/Result pairs, so long as the chosen Steps meet all the necessary and sufficient conditions of the USF logic tree.

Appendix F

Flying Logic Software Automates USF Logic Trees (ref: S&T Tree Pp 77-82)



FLYING LOGIC

Thinking with Flying Logic

by
Robert McNally

Version 1.0.1



Documentation © 2011 Sciral



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License](https://creativecommons.org/licenses/by-nc-nd/3.0/).

Sciral
726 E. Colorado Ave. #9
Glendora, CA 91740
FlyingLogic.com

Contents

Part I — Introduction	5
About This Book	5
Keys to Great Thinking	7
Part II — The Theory of Constraints Thinking Processes	11
Overview of the Theory of Constraints	13
The Goal	13
The Constraint	13
The Five Focusing Steps	14
The Categories of Legitimate Reservation	17
Clarity	17
Entity Existence	20
Causality Existence/Cause-Effect Reversal	20
Insufficient Cause	23
Additional Cause	24
Predicted Effect	
Tautology	
Current Reality Tree	25
Evaporating Cloud — Conflict Resolution	35
Future Reality Tree	43
Prerequisite Tree	55
Transition Tree	61
Strategy & Tactics Tree	69
Part III — Other Techniques	75
Evidence-Based Analysis	77
Concept Maps	81
Appendix	83
Resources	83
Flying Logic Web Site	83
Web Sites on the TOC	83
Books on the TOC	83
Books on Psychology, Communication, and Negotiation	84
Other Useful Web Sites	84

Part I — Introduction

About This Book

Flying Logic is software that *helps people improve*. This book, *Thinking with Flying Logic*, introduces the core techniques that the Flying Logic was designed to support. Even if you don't use Flying Logic, I hope you will find it a concise and useful introduction to some powerful ways you can improve your business and personal life.

Thinking with Flying Logic is companion to two other documents: *Welcome to Flying Logic* explains why Flying Logic exists, and the *Flying Logic User's Guide* explains the details of operating it. To use a travel analogy, *Welcome to Flying Logic* hopefully got you interested in taking a trip, the *Flying Logic User's Guide* taught you how to drive the car, and *Thinking with Flying Logic* is the road map you will follow to get you where you want to go.

However, *Thinking with Flying Logic* is not an exhaustive tutorial on the techniques it discusses— in fact, it barely scratches the surface. In particular, the Theory of Constraints (TOC) and the TOC Thinking Processes that inspired the creation of Flying Logic are supported by a wealth of literature, books, papers, web sites, courses, conferences, consultants, trainers, academics, implementors, studies, and success stories. I believe that Flying Logic is a much-needed piece of the puzzle, and I urge anyone who reads this book to seek out these other great resources as well, some of which are listed in the [Appendix](#).

Keys to Great Thinking

Most of this book is spent on the step-by-step instructions for working with each of the techniques it presents, but in this introduction I want to briefly touch on some ideas, attitudes, and behaviors that I have found create a mind set conducive to effective thinking and communication—these are the ultimate keys to effective use of Flying Logic.

Logic and Emotion

“Logic” is popularly seen as a cold, complex topic; on par with higher mathematics and invoking images of nerdy professors, science fiction computers and emotionless aliens. But the fact remains that *we all think*, and we all use logic with more or less skill.

What is not widely understood is that logic is simply the *rules for thinking*. Just as it is possible (though perilous) to drive a car without knowing the rules of the road, it is possible to think without understanding the rules of logic. These rules are extremely powerful, and fortunately quite simple— but it is unfortunate that as children we are rarely taught to use them as naturally as we learn to read and write. And far from turning us into dispassionate machines, we humans are naturally the happiest and most productive when our emotional hearts and logical minds work together in concert.

Some people resist “being logical” on the grounds that they “just know how they feel” on a given subject. But when we experience strong emotions or gut instincts, it is important to recognize that there are always underlying *causes* for those feelings. If we merely acknowledge the resulting feelings, and resist a deeper understanding of the causes, we create a disconnect between the rational and emotive parts of our minds. This disconnect results in *cognitive dissonance*, which is stress resulting from attempting to believe conflicting things or behave in conflicting ways. Cognitive dissonance is a two-edged sword: on the one hand it can help motivate us to change our beliefs for the better (that is, to better reflect reality) while on the other hand it can also lead us to manufacture *rationalizations* for the way we feel that don’t reflect reality. While both actions quell the discomfort of cognitive dissonance in the short term, rationalizing ultimately leads us deeper into trouble by putting us further and further out of sync with reality.

Attempting to act on feelings alone has another drawback: such actions

leave us vulnerable to unintended consequences that our rational minds could have helped us predict and avoid. Of course, it works the other way too: if we try to be “purely rational,” yet ignore strong feelings *by discounting their causes*, we are also going to create dissonance.

The solution is to get in the habit of bringing the causes (or *reasons*) that underlie our emotions and instincts to the surface. In doing so, we validate our emotions, and can then integrate them into effective plans.

The good news is that thinking is a *learnable skill* that improves with practice, and that doing so does not diminish, but rather *complements* the value of emotions.

Communication and Criticism

We can rarely accomplish anything of significance alone: we rely on other people for many kinds of contributions, and since no one is an island, we must communicate effectively with others— to gain an understanding of their needs, benefit from their experience and wisdom, and negotiate their cooperation.

Often, we are too close to a situation to understand it well— we are embroiled in the situational details and “can’t see the forest for the trees.” When we think we understand a situation well; when we think we already know all the options and the right answers— this is when inviting others to evaluate and criticize our plans can be the most valuable. Doing so lets “light and air” into our minds and helps us rid ourselves of ways of thinking that have become stale and unproductive.

In *The Godfather Part II*, Michael Corleone says, “Keep your friends close, but your enemies closer.” Ironically, the most fruitful criticism often comes from people who actively disagree with us. Abraham Lincoln, arguably the greatest United States President, is renowned for having chosen prominent members of his cabinet from those who most vehemently opposed his policies. Whether or not we ultimately agree with our critics, they can often teach us a great deal— the key is to allow our view of the world to change as we learn.

Argument and Honor

When we think of an argument, many of us envision scowls, angry gesticulation, and yelling. We imagine petty name-calling, a parade of unforgiven grievances, and other emotional power plays. Most importantly, we imagine arguing to *get our way*— to show that we are *right*

and others are *wrong*. But such an interaction is not an argument— it is a *fight*. In a fight there may be winners, but there will certainly be losers, and injuries for all.

A real argument is a *shared search for truth*. In an honorable argument people can still be passionate, but they follow the rules of logic just as drivers follow the rules of the road. And even though people approach a situation from different perspectives and with different preconceptions, the positions they take should be seen as suggestions that are *ultimately intended* as win-win, even if they initially fall far short. Indeed, even such flat statements as, “We’ll get along fine as soon as you learn to do things my way,” hint at a common objective: *getting along*.

When argument is viewed as a search for truth, it becomes possible to see adapting one’s position to new information and ideas not as weak or wishy-washy, but as a challenge to which only a mature, strong, and honorable person can rise. More pragmatically, all sides can begin to look forward to not merely *getting their way*, but *getting something better* in the form of a win-win solution.

Control and Influence

When considering how to cause change, we can imagine ourselves standing at the center of a circle. The things we can reach out and touch directly define our *span of control*. If the all changes we wish to make are entirely within our span of control, we have the power to simply go ahead and make them.

Usually, however, things are not so simple. In our mental image, the things we control are just what lies within arm’s reach— our span of control is always quite small. But just beyond our span of control lies the start of our *sphere of influence*. Although we may not be able to reach out and touch these things directly, we can still cause change by cooperating with others. For example, a business may *control* its manufacturing processes, while it can only *influence* its suppliers and customers.

The farther away objects are, the less influence we wield— until we reach a point where we have no significant influence. This marks the end of our sphere of influence.

Our sphere of influence is always much larger than our span of control, and is probably larger than we think. Most gratifyingly: causing positive changes within your sphere of influence has the desirable effect of expanding it.

Optimization and Suboptimization

When we reward people for improvements entirely within their *span of control*, what is the natural reaction? An example of this might be basing manager performance reviews solely on efficiency within their departments. The natural reaction is, of course, for them to *narrow their span of control* as much as possible— to define its boundaries as sharply from other parts of the system, and to focus entirely on efficiency within their particular component (division, department, cubicle, etc.) This behavior results in *suboptimization*, which is maximizing or fine-tuning a part of the system without considering the (often detrimental) effects of doing so on the entire system.

On the other hand, what happens when we reward people for improvements within their entire *sphere of influence*? In this case, their desire becomes to extend their sphere of influence outwards as far as possible. As mentioned previously, acting in one's sphere of influence requires coordination and cooperation with others, which in turn encourages an awareness of the system as a whole. The end result is *optimization*, where people orchestrate their efforts together, toward the fulfillment of the system's goal.

Optimization is the outcome of *systems thinking* (looking at a system not as merely a collection of parts but as a unified whole) applied to the goal of *process improvement*.

Tools and Expectations

People have invented many useful tools that help us perceive the world accurately, arrange our knowledge, think about it logically, develop plans, and communicate effectively. Despite having these tools, we must still do the hard work of thinking, and also the hard work of implementing our plans. When new tools (such as Flying Logic) are introduced, they are often touted as labor-saving devices. But do we really do *less* work now that we have automobiles, telephones, and computers? Arguably, in our world of accelerating change, we often do *more*. So it is important to have a pragmatic understanding that the net result of new tools is not to *reduce labor*, but to *raise expectations*.

Just as spreadsheets were a boon to accounting and financial planning but did not make accountants obsolete, I hope that Flying Logic will be of significant help to systems thinkers and people with a passion for making the world and its systems better. Even more, it is my hope that Flying Logic will help get more people involved in these vital topics.

— Robert McNally, 2007

Part II – The Theory of Constraints Thinking Processes

Overview of the Theory of Constraints

The Goal

The [Theory of Constraints](#) (TOC) is an overall management philosophy originally developed by [Eliyahu M. \("Eli"\) Goldratt](#) and first popularized in his bestselling business novel [The Goal](#). He started with the idea that all real-world **systems**; whether personal, interpersonal, or organizational have a primary purpose, or **goal**. The rate at which the system accomplishes its goal is called **throughput**.

The Constraint

From the idea of throughput, it is easy to see that systems must also have at least one **constraint**: something that limits the system's throughput, which can be likened to a chain's weakest link. If a system had absolutely no constraints, it would be capable of infinite throughput. But though infinite throughput is impossible, amazing throughput gains *are* possible through the careful identification and management of a system's key constraints. The purpose of the TOC then, is to give individuals and organizations the tools they need to manage their constraints in the most effective manner possible.

Originally applied to industrial manufacturing lines, TOC principles have been successfully adapted for areas as diverse as supply chain, finance, project management, health care, military planning, software engineering, and strategy.

TOC claims that a real-world system with more than three constraints is extremely unlikely, and in fact usually only one constraint is key. Counter-intuitively, this is because the *more* complex a system becomes, the *more* interrelationships are necessary among its parts, which results in *fewer* overall degrees of freedom.

A major implication of this is that managing a complex system or organization can be made both simpler and more effective by providing managers with few, specific, yet *highly influential* areas on which to focus — maximizing performance in the areas of key constraints, or **elevating** the constraint (making it less constraining.)

The TOC was originally applied to manufacturing operations, where the

constraint was usually a **physical constraint**— some sort of machine or process that formed a bottleneck in the production line. These sort of constraints are fairly easy to locate. But in the real-world situations where these constraints were **broken** (i.e. elevated to the point where they were no longer *the* constraint) it was discovered that the constraints could take on another character: the **policy constraint**. These are the “ways things have always been done” that ultimately serve to restrict the system’s throughput, and they are usually due to some form of **suboptimization**— tuning part of a system without regard to the benefit of the whole. Policy constraints are often more difficult to identify and more difficult to manage than a simple machine or physical process— more powerful tools were invented to do just that.

The Five Focusing Steps

To identify and manage constraints of all kinds, the developers of TOC defined the **Five Focusing Steps**, which describe a process of ongoing improvement. (Step Zero was later added for additional clarity.)

0. **Articulate** the goal of the system. *How do we measure the system’s success?*
1. **Identify** the constraint. *What is the resource limiting the system from attaining more of its goal?*
2. **Exploit** the constraint to its fullest. *How can we keep the constraining resource as busy as possible, exclusively on what it can do that adds the most value to the entire system?*
3. **Subordinate** all other processes to the decisions made in Step 2. *How can we align all processes so they give the constraining resource everything it needs?*
4. **Elevate** the constraint. *If managing the constraining resource more efficiently does not give us all the improvement we need, then how can we acquire more of the resource?*
5. **Avoid inertia**. *Has the constraint moved to some other resource as a result of the previous steps? If so, don’t allow inertia itself to become the constraint: go back to step 1.*

It is possible that, after iterating through the Five Focusing Steps a few times, that the constraint on the system’s throughput moves entirely out of the system itself, and into the system’s environment. An example of this would be when a manufacturer has more capacity than demand for its products. In this case, further improvement may still be possible,

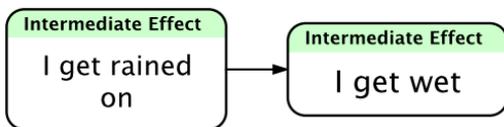
but doing so requires expanding the concept of the “system” to include its customers, the economy, and other factors that were originally just givens of the system’s environment.

The Thinking Processes

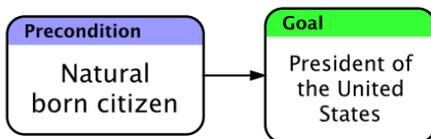
The **Thinking Processes** emerged as TOC practitioners worked with organizations that needed to identify their core constraints and how to manage or elevate them. They needed the answers to three deceptively simple questions:

- **What** to change?
- **To what** to change?
- **How to cause** the change?

The Thinking Processes are based on the scientific method, to which is added a simple visual language, the **Thinking Process Diagrams**, that are used for describing and reasoning about situations, arguments, and plans using the language of **Cause and Effect**. There are two basic kinds of reasoning: **Sufficient Cause** and **Necessary Condition**.



A sufficient cause for an effect



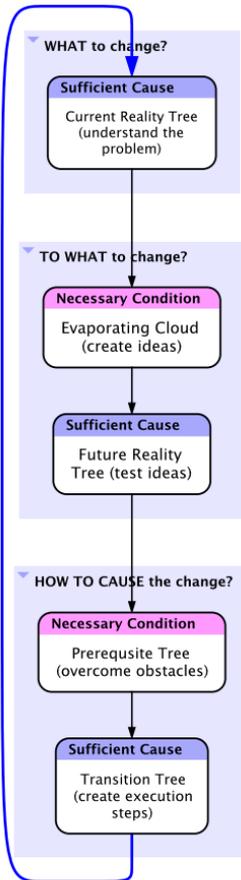
A necessary condition for an effect

The Thinking Process Tools

From the basic Thinking Processes developed several techniques called the **Thinking Process Tools** designed to answer the three questions. The tools provide the ability to develop a complete picture of a system’s core constraints and how to manage them.

Tool	Thinking Process	Starting Point	End Result
Current Reality Tree (CRT)	Sufficient Cause	A set of undesirable symptoms	The core cause of the symptoms (constraint)
Evaporating Cloud	Necessary Condition	A perceived conflict underlying a constraint	Possible win-win solutions
Future Reality Tree (FRT)	Sufficient Cause	A proposed solution	Necessary changes that implement the solution and avoid new problems
Prerequisite Tree (PTR)	Necessary Condition	Major objectives and the obstacles to overcoming them	Milestones that overcome all obstacles
Transition Tree (TRT)	Sufficient Cause	A set of goals	Detailed actions to achieve the goals
Strategy & Tactics Tree (S&T)	Necessary Condition	The highest-level goals of a system	A multi-tiered set of implementation steps

The last of these tools— the **Strategy & Tactics Tree**, is used in large organizations where it is necessary to create major changes in a short period of time. However, the other five tools are applicable to systems of any size from individuals, to families, to businesses small and large. Like a physical tool kit, you can choose to use individual tools— just the right tool for the job at hand. Or, you can do a larger project where most or all of the tools may be required. When all of the tools are used, the “finished result” of one tool can easily be used as part of the “raw materials” for the next tool. Since improvement is a continuous process, you can use each tool over and over again on every pass through the Five Focusing Steps.



The Measurement of Success

The last piece of the improvement puzzle is *feedback*. There needs to be an unambiguous way to measure improvements brought about through the implemented changes. For traditional business, Dr. Goldratt developed three *non-traditional* measurements that began with the overriding concept of the system's *goal*: **Throughput (T)**, **Inventory (I)**, and **Operating Expense (OE)**. It is outside the scope of this book to discuss these in detail, but readers are directed to the TOC body of knowledge (see the [Appendix](#)) for discussions of these measures and how they have been adapted for many different endeavors.

The Categories of Legitimate Reservation

We all want our ideas and plans to make sense. But how do we *know* that we are making sense? What do we even mean by that? When we use the **Thinking Process Tools**, we are building a model of the way part of the world works, and in this context our model *makes sense* if it in fact portrays a picture of the world that is *pertinent* and *accurate*.

To be *pertinent*, our model must be of that part of the world (our system) that we actually care about— in other words our model must have the proper *scope*. It must not be too detailed in areas that don't significantly affect the outcome, nor too general— glossing over areas where important details lie. To ensure pertinence, the people who are the main stakeholders in the outcome of the plan must have influence over it.

To be *accurate*, the cause-and-effect relationships that we model must indeed hold in real life. The **Categories of Legitimate Reservation (CLR)** are ways to verify the accuracy of a **Thinking Process Diagram**. They are used to catch common pitfalls in our own thinking and the thinking of others. They are called the *Categories* because they are well-defined and of limited number. They are called *Legitimate* because anyone who writes or reads logical statements is always allowed to express them. And they are *Reservations* because they highlight parts of the diagram that are not completely convincing. Since these reservations are *always legitimate*, they can be raised, explored, understood, and accepted without anyone feeling like they're having their toes stepped on— they help everyone keep their emotional distance and stay reasonable.

When you start to work with Thinking Process diagrams, you should deliberately consider the CLR one by one for each part of your diagram. But as you gain experience you will find you begin to apply them quickly and habitually.

Clarity

If you are creating a Thinking Process diagram by yourself, you probably have a good idea of what you mean. However, you will also probably need to share your plan with someone else sooner or later, and you

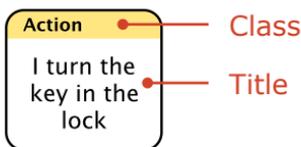
need to apply the Clarity reservation as the *last* step before you do. Ask yourself:

- Is the meaning of each part of my diagram clear?
- Is the meaning of my diagram as a whole clear?

Similarly, when someone presents you with a Thinking Process diagram you have never seen before, you should apply the Clarity reservation *first* by asking yourself:

- Does this diagram really convey what the person presenting it intends?

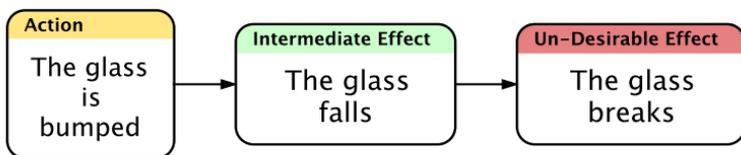
In Thinking Process diagrams, causes and effects are all represented by **entities**: rectangles that contain brief statements that are, or could be, true about reality. Flying Logic entities also have a colored bar at the top that designates the entity's **class**— the kind of role the entity plays in the diagram of which it is part.



To satisfy the clarity reservation, the **title** of an entity must be:

- complete, unambiguous, and grammatically correct,
- in the present-tense, and
- *simple* in that it contains a single idea with no compound statements.

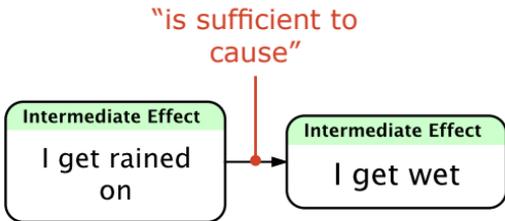
“Bumped and glass fell and broke,” is an example of a statement that violates all three principles. This idea should probably be expressed as three separate entities, each related to the next by a causal connection:



The causal connections between the entities must also be clear, with each step from entity to entity having a natural and obvious flow for any stakeholder who reads the diagram. Reading from one entity to another via an **edge** (also called an **arrow**) will follow one of two patterns, or **Thinking Processes**. Which Thinking Process is used depends on what kind of diagram you are working with; but within a single diagram, the meaning of the edges does not change.

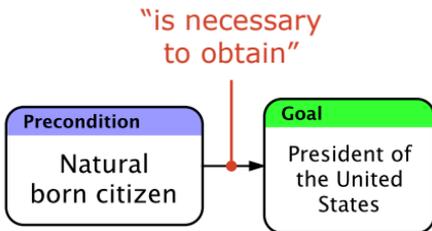
- **Sufficient Cause Thinking:** "If **A** then **B**." or "**A** is sufficient to cause **B**."

This pattern expresses the idea that the existence of **A** is, by itself, enough to cause the existence of **B**. Sufficient Cause Thinking is used by the **Current Reality Tree**, **Future Reality Tree**, and **Transition Tree**.



- **Necessary Condition Thinking:** "If not **A** then not **B**." or "**A** is necessary to obtain **B**."

These patterns express that **A** must exist for **B** to exist, but may not be sufficient by itself. Necessary Condition Thinking is used by the **Evaporating Cloud** and **Prerequisite Tree**.

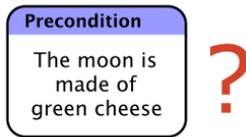


Notice that in both illustrations, the edge (arrow) looks exactly the same

although the meaning is different. How you read an edge depends on which Thinking Process was used to construct the diagram.

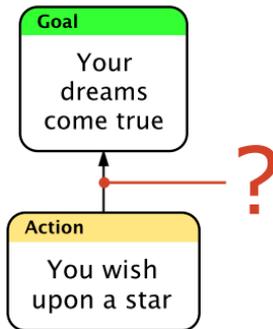
Entity Existence

This reservation asks whether an entity in the diagram *is true now*. In a Current Reality Tree, for instance, every entity in it should describe something that *is true now*. A Future Reality Tree or Transition Tree, however, can contain a mix of entities that are either true now, or would be expected to become true under certain conditions. This reservation is a warning to “check the facts” before making an untrue assertion about reality.



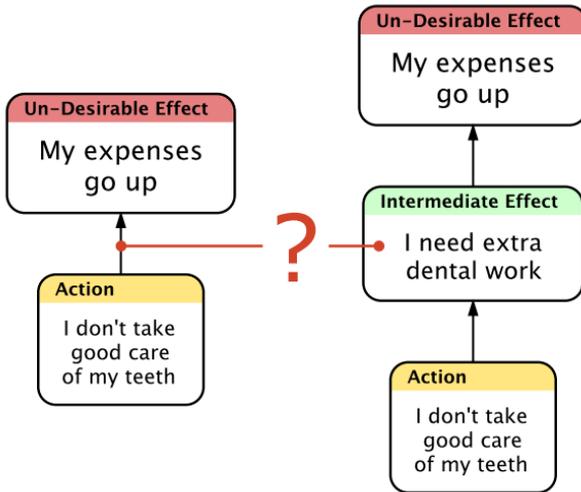
Causality Existence/Cause-Effect Reversal

This reservation asks, “Does **A** really cause **B**?” Often we associate two ideas because they are *correlated*, that is, they are often found in proximity to each other. However, to actually say that one thing *causes* another requires much stronger evidence.



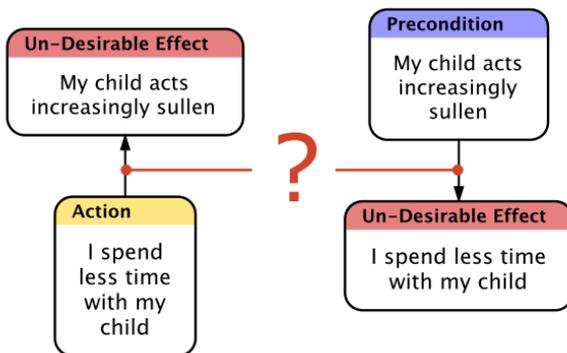
Indirect Effects

Other times, an entity is an **indirect effect** of a cause, but important necessary steps are missing.



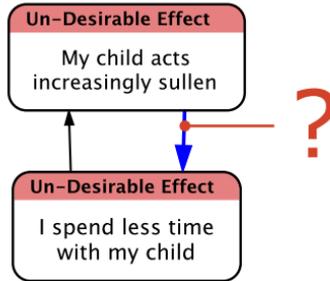
Cause-Effect Reversal

A special case of the Causality Existence reservation is **Cause-Effect Reversal**. In this case, we question whether the edge is pointed in the right direction.



Back Edges

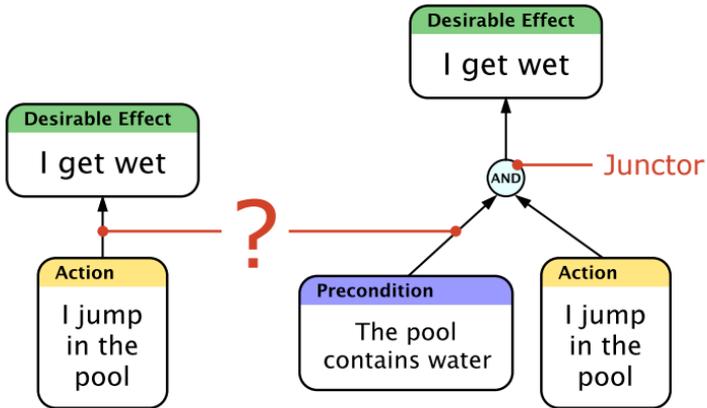
In cases where it seems ambiguous as to which entity is the cause and which is the effect, it may be a good place to look for a self-reinforcing loop. Flying Logic can model self-reinforcing loops using **back edges**. A back edge is added whenever you attempt to create a new edges that indirectly makes an effect to be its own cause. Back edges are drawn thicker than regular edges.



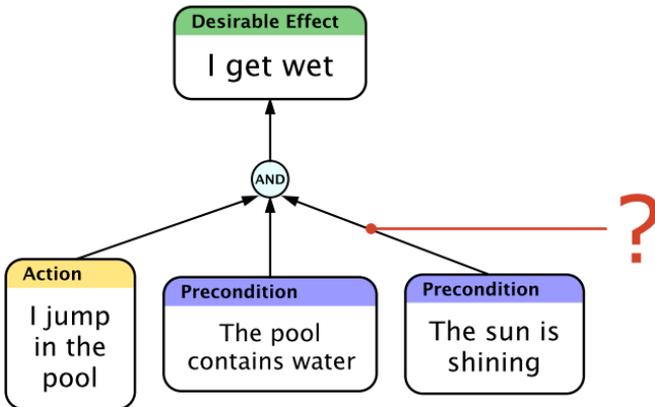
Insufficient Cause

This reservation asks, "Is **A**, all by itself, sufficient to cause **B**? What else might also be necessary?" Usually a combination of factors outside our control ("Preconditions") and factors that we influence or control ("Actions") must combine to create a particular effect. In diagrams based on Sufficient Cause Thinking, this is modeled using a **junctor** that

contains the **AND** operator. Junctors are easily created by dragging from an entity to an existing edge.

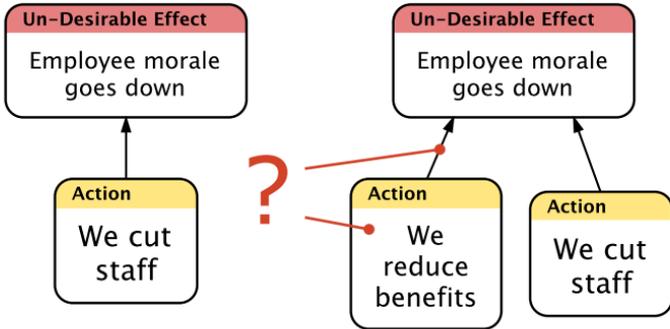


When looking for insufficient causes, we should also keep in mind that a list of causes can also be *too sufficient*, or in other words, include causes that are actually not required to produce the effect. So we should also ask, "Have we listed anything as necessary that really isn't?"



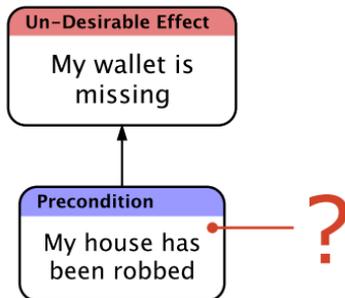
Additional Cause

Once we have identified one sufficient cause for an effect, we are often tempted to move on, and in doing so we may overlook other causes that may either be independently causing the effect, or mutually intensifying it. This reservation asks, "Have we identified every cause of **A**? What else could also be causing **A**?"



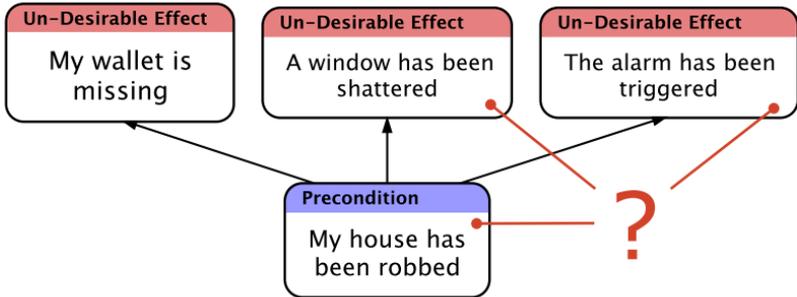
Predicted Effect

How can we increase our certainty that a cause we have identified is *really* the cause of the effects we are inclined to believe? For example, let's say I come from a walk and discover my wallet missing. One of the first things that might pass through my mind is that my house has been robbed. But *has* it been robbed?

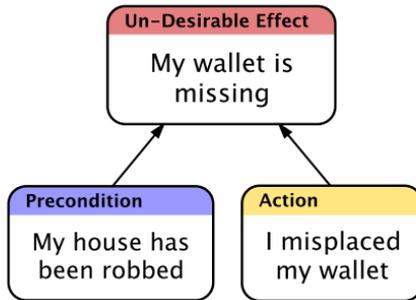


Usually a cause is responsible for more than one effect, and this reser-

vation asks, "If **A** is true, what other effects *in addition to B* would we expect to see?"



If the additional predicted effects are also observed, then we can be more confident in the causality we initially identified. But if the predicted effects are *not* observed, then we may be well advised to look for additional causes.



Tautology

People sometimes don't examine their beliefs very closely, and will, when asked for a cause, often re-state the cause using different words. Even though you will almost never encounter tautology (also called *circular reasoning* or *begging the question*) in a Thinking Process diagram, you will encounter it in casual conversation. Some examples:

- “You can’t give me a C for this course— I’m an A student!”
- “My homework is boring because it’s so tedious.”
- “Mayor Green is the most successful mayor ever because he’s the best mayor in our history.”
- “The defendant shows no remorse, and this fact should strengthen your resolve to find him guilty!”

Current Reality Tree

When a non-trivial system (a for-profit business, a non-profit organization, a department, or a personal relationship to name a few examples) needs improvement, it is often not clear *what to change*, even to people who have a great deal of experience with the system's workings. This is because systems contain many cause-effect relationships that interrelate in complex ways, and understanding the system sufficiently to decide what to change is often even more problematic because the people with experience often have only a narrow view of the parts of the system they interact with.

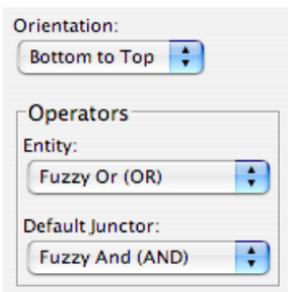
The Theory of Constraints (TOC) is based on the idea that all systems have a *goal*, or reason for existence—the rate at which a system can achieve its goal is called its *throughput*. The TOC also says that all systems have *core drivers*, which can be physical constraints, policy constraints, market constraints, or some combination of those, that have a major impact on the entire system and that ultimately (albeit indirectly) govern the system's throughput. Ironically, the *more* complex the system, the *fewer* core drivers it is likely to have, due to the greater number of interdependent cause-effect relationships such systems contain.

The **Current Reality Tree** (CRT) is a tool for discovering the system's core driver, which is also known as *the constraint*. The constraint is the cause that is *most common* to the *most severe* symptoms the system is experiencing, and thus the constraint must be managed most carefully in order to most dramatically improve throughput. By focusing on the constraint, you will realize the most “bang for your buck.”

Flying Logic Setup

A CRT is based on [Sufficient Cause Thinking](#), and this is how Flying Logic documents are set up when first created, so you do not need to do anything special with the Operators popup menus to start creating your CRT. Most CRTs are drawn with root causes at the bottom and the

symptoms at the top, so you may want to use the Orientation popup menu to change the orientation of your document to **Bottom to Top**.



CRTs are created using the entity classes in the built-in Effects-Based Planning domain, and primarily use the following classes: Un-Desirable Effect, Precondition, and Intermediate Effect. CRTs are most often used to pinpoint problems, but can also be used to identify core strengths, in which case the Desirable Effect class can also be used.

- Effects-Based Planning
- Goal
- Intermediate Effect
- Precondition
- Action
- Un-Desirable Effect
- Desirable Effect

Step 1: Understand the Scope

Before you can document how your system works and where its problems lie, you need to make sure you have a clear understanding of what you mean when you talk about your system. In other words: what are you analyzing?

Spend the time necessary to reach a clear, written understanding with other stakeholders:

- What is your system's goal?
- What are the necessary conditions for knowing the goal is being

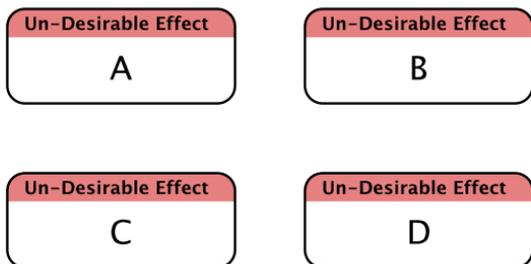
achieved?

- What measures do you use to use to know how well the necessary conditions of the goal are being met?
- Where do the boundaries of your system lie?
- What greater system is your system a part of?
- What systems does your system interact with?
- What are your system's inputs and outputs?

Step 2: List the Symptoms

Presumably, you are doing your analysis because you believe the system would benefit from improvement, and because you see evidence of this potential benefit in various problems or symptoms of trouble. Such symptoms could be low profits, low customer satisfaction, or lots of arguments among family members. These symptoms are known in TOC as **Un-Desirable Effects** or simply **UDEs**.

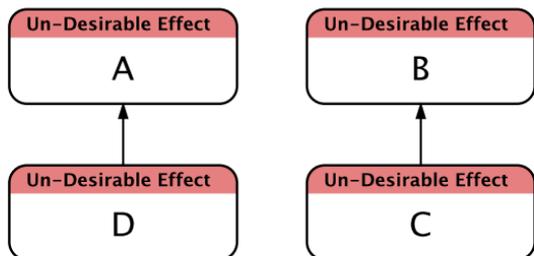
Usually there are between 5 and 10 UDEs that are causing the most difficulty in the system, and it is these UDEs that should be added first. Give each UDE a simple, present-tense title that is intended to be clear to any stakeholder, and make sure that the UDEs you choose at this stage are uncontroversial as to their actual existence. In other words, any stakeholder who looks at this list should have no difficulty agreeing, "Yes, these are some of the most serious problems we have."



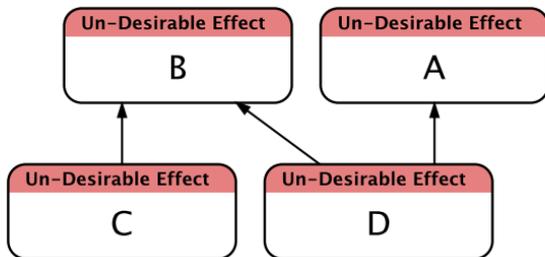
Step 3: Connect the Symptoms

Undesirable Effects often contribute to other problems. As you study your list of UDEs, you will notice that some are probably direct or indi-

rect causes of others already in your list. If this is the case, then connect these entities with *edges* (arrows) from the causes to the effects. Don't be too concerned at this stage if the causes are not directly responsible for the effects: as you grow the tree, you will add other entities that complete the picture.

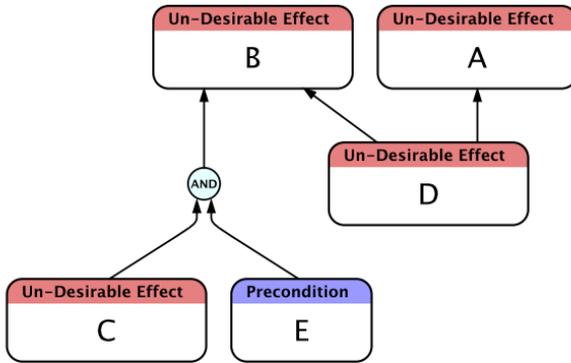


Sometimes you will notice that a single cause contributes to more than one effect, as is the case with **D**, below.



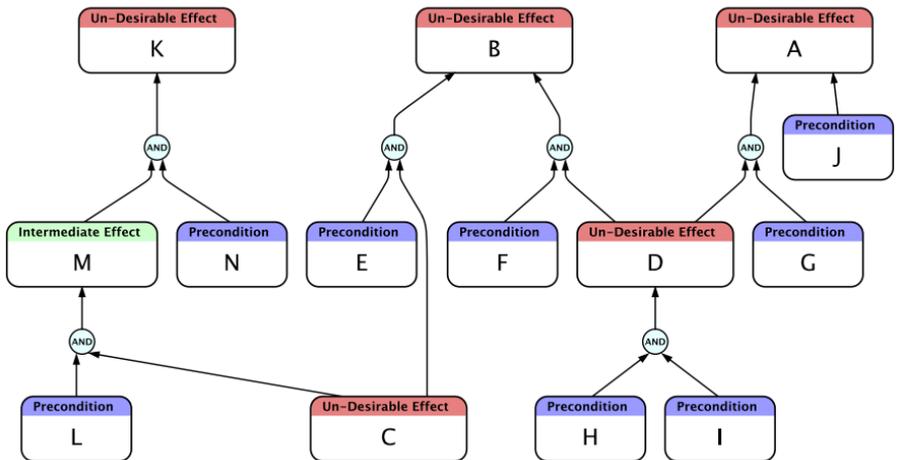
Other times you will notice that an effect has more than one independent cause, as is the case with **B**, above. When a Flying Logic document is set up for Sufficient Cause Thinking, more than one arrow entering an entity denotes more than one *sufficient, independent cause*. This is also called an **OR** relationship.

Often, a single cause is *necessary, but not sufficient* by itself to cause an effect. This is denoted by an **AND** junctor, which is created by dragging from the cause entity to an existing edge.



Step 4: Apply the Categories of Legitimate Reservation

The diagram as it stands is probably only an extremely rough picture of your system. By applying the [Categories of Legitimate Reservation](#), you now add additional entities and causal relationships that create a true picture of the situation. In particular, look to add *additional causes* for the effects you have identified, and identify *insufficient causes* and add their *necessary conditions*. Also review your diagram for *clarity*, and step through it using Flying Logic's confidence spinners. You can even change the class of an existing entity if, for instance, an entity that you originally added as an UDE now appears more neutral in context.

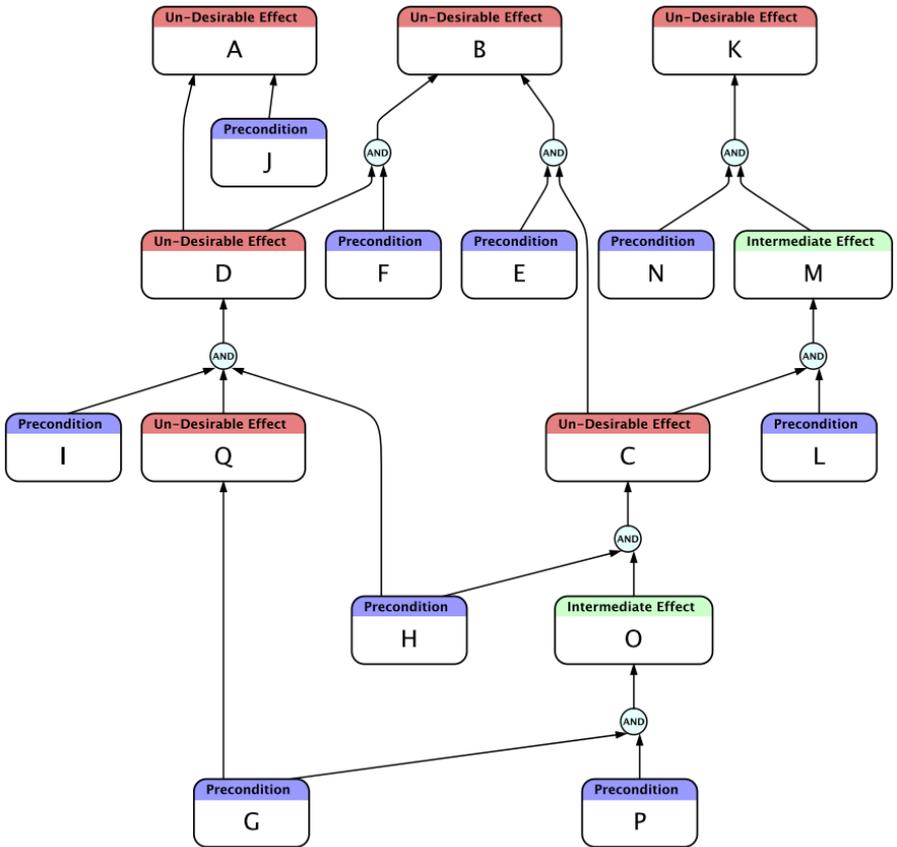


Use these guidelines to help you choose what class of entity to add:

- If an entity is undesirable on its face— in other words, if the system would definitely be better off without it, then use the **Undesirable Effect** class. UDEs can have predecessors, successors, or both, but should always have at least one causal connection into a completed diagram.
- If the entity is neither negative nor positive, but exists merely due to the larger context in which the system must operate and is something over which you have no significant influence, then use the **Precondition** class. Preconditions should never have predecessors, and should always have at least one successor.
- If the entity is neither negative or positive, but exists because of something within your control, then use the **Action** class. Actions are always causes and never effects, so they will have successors but no predecessors.
- If the entity is neither negative nor positive, but it exists as a consequence of other causes in the diagram, use the **Intermediate Effect** entity class. In a completed CRT, Intermediate Effects should *always* end up with both predecessors and successors.

Step 5: Continue Adding Underlying Causes

At this stage, you may have several unconnected, or loosely-connected clusters of entities. In this step, search for and add deeper causes for the effects in your diagram, looking in particular to add causes that tie two or more clusters together. Of the causes that are currently at the root of your diagram, keep asking yourself, “Why is this happening?” and make your answer take the form of additional entities and the edges that connect them. Alternate between adding underlying causes and applying the Categories of Legitimate Reservation from the previous step.

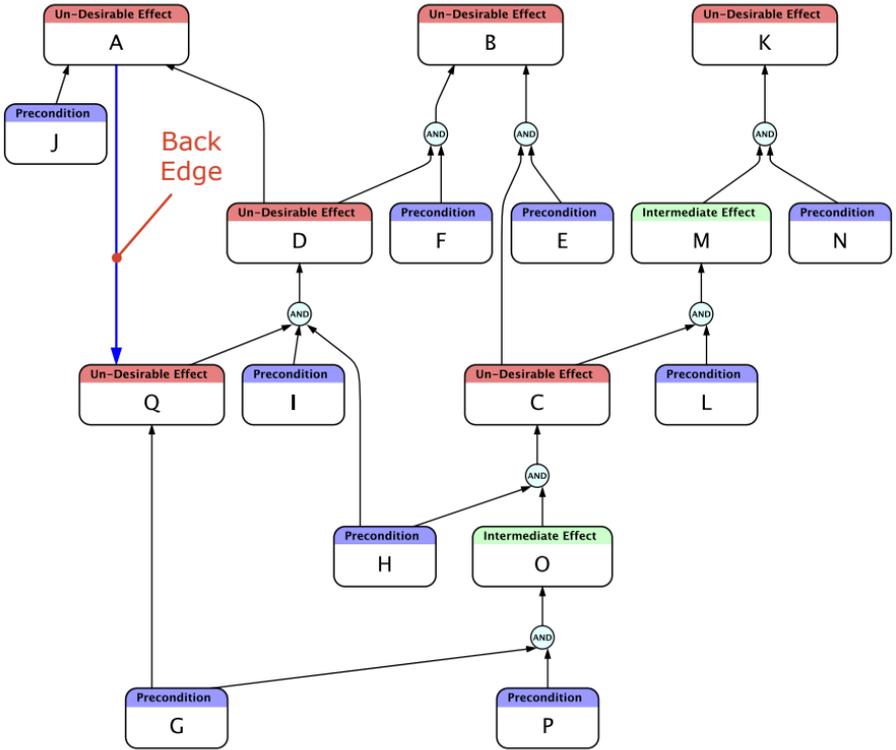


Step 6: Consider Negative Reinforcing Loops

Although it is uncommon, sometimes the presence of an UDE at a higher level in your system actually aggravates UDEs at a lower level. Since this situation is so serious, it is important that you note it in your diagram as a **causal loop**, also known as a vicious circle.

Normally, all edges in a Flying Logic document “flow” from the start of the document (the root causes) to the end (final effects.) When you attempt to add an edge that would create a loop (that is, an effect indirectly becoming its own cause) Flying Logic creates a special **back edge** that denotes a causal loop. Back edges are thicker than regular edges

and drawn in blue.



Back edges differ from regular edges in two ways: They do not have edge weights, and they do not participate in the flow of confidence values through the documents. They can, however, be annotated like other edges.

Step 7: Identify Root Causes

As your CRT becomes more complete, you will notice that one group of causes lie at its "root." That is, they have successors but no predecessors. Some of these causes will be Preconditions and others may be UDEs, and they won't necessarily appear at the bottom of the diagram—in the illustration above there are nine root causes.

The purpose of this step is to make sure you have built down your tree to the point where you have uncovered the deepest causes *over which you have some control or influence*. Preconditions are by definition out of our control, but you should question whether the preconditions at the root of your CRT aren't really Intermediate Effects with other underlying causes. Also question whether the UDEs at the root of the tree don't have additional underlying causes that you also control. The idea is to "uproot" problems at their deepest possible level.

Step 8: Trim the Tree

At times you may discover that parts of the tree you have built have little or no connection, as successors or predecessors, to the UDEs you care about. To keep the tree manageable, you should remove these clusters from view by either

- Deleting them,
- Using Cut and Paste to move them to a different document, or
- Placing them into a group which is then collapsed.

Step 9: Identify the Core Driver

If you have constructed your CRT rigorously observing the rules of cause and effect, you will agree that eliminating a root cause will also cause a chain reaction of other problems being eliminated. If this doesn't appear to be the case, go back and make sure that at each step in the diagram, you have identified and added all the *necessary* and *sufficient* causes of your UDEs (Step 4), and that you have built the tree down as far as possible to root causes that you control or influence (Step 5.)

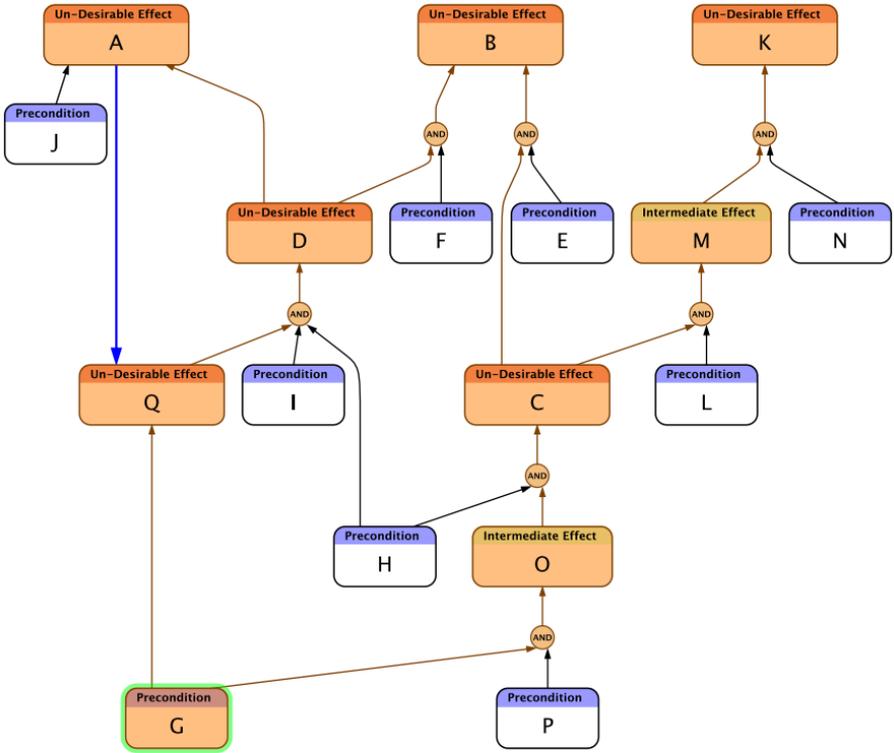
The time has come to identify the single cause that has the most influence over the most critical UDEs in your CRT. This single cause is the Core Driver (also call the *constraint*, or the *bottleneck*)— the cause that must be managed or eliminated in order to break through the boundaries that hold your system back.

Although your CRT may contain several root causes, all of which may eventually need attention, you can find the Core Driver by judging several factors for each root cause:

- **How many** UDEs they indirectly cause,
- **How severe** are the UDEs they indirectly cause, and
- **How much control or influence** you have over them.

PRO If you use using Flying Logic Pro, you can turn on the **Edit ↗ Cut/Copy Includes Successors** switch, select one of your root causes, then select **Edit ↗ Copy**. This highlights the cause you selected, *and* all of its direct and indirect predecessors. (Press the **Escape** key to remove the highlighting.) Use this technique to quickly get an idea of how influential each of your root causes are, although you will still need take the severity of the UDEs into account.

In the illustration below, entity **G** has been selected and copied with the **Includes Successors** switch on, which makes it obvious that it contributes in some way to every UDE in the diagram. Since all of the causes are at least within our influence, we conclude that **G** is our Core Driver— it is the constraint on which we must focus.



Evaporating Cloud – Conflict Resolution

Arguments. Fights. Politics. Enemies. Compromise. Loss.

We have all encountered conflict, and most of us try to avoid it whenever possible. Conflict is seen as unhealthy and unpleasant to the point that many people will attempt to ignore it even after realizing that doing so may actually be contributing to a worsening situation. Or both sides treat the conflict as a zero-sum game: “Either I go, or he goes!” Or perhaps worse, both sides compromise: they “split the baby” and nobody goes away happy.

It turns out there are better ways for resolving conflicts: ways that result in the creation of solutions that completely satisfy everyone involved. From one remarkable perspective, it is even possible to entertain the idea that *conflicts don't actually exist* except at the superficial level of our *positions*: what we say we *want*.

When two wants appear to be mutually exclusive, we say there is a conflict. The way forward is to recognize that our *wants* (also called *positions*) are motivated by underlying *needs* (also called *requirements*.) For example, the two wants could result from children fighting over a toy: in this case they both *want* to possess the same limited resource. However, they are motivated by underlying *needs*, which may not be the same for each of them: one child may feel the need to assert their ownership of the toy, while the other child may feel the need to incorporate the toy into their play. Furthermore, the children are united in a *common objective*: to get along and have fun. To achieve this common objective, satisfying both children's needs is necessary. Notice that as we have passed beyond the boundary of the apparent conflict presented by their wants, a recognition of their needs and common objective begins opening the door to creative solutions that may leave everyone happier than they thought possible.

When the connection is made that conflict stems *not* from some kind of pathology, but from *legitimate needs* and *common objectives*, it becomes obvious that the best approach is not avoidance but prompt communication and the creation of options for mutual gain.

Often, after producing a Current Reality Tree (CRT), it is possible to recast the Core Driver as a *Core Conflict*, containing two mutually ex-

clusive positions. So whenever we find ourselves faced with conflicting wants, which often happens as the result of creating a CRT, but even more frequently just happens on its own, the **Evaporating Cloud** is the tool to use. (It is so-called due to its ability to “evaporate” conflict. It is also known as the **Conflict Resolution Diagram**.)

Flying Logic Setup

Since the basic form of a Cloud is always the same, the easiest way to start a Cloud is to open the included template file **Cloud.logic-t** located in the **Examples/Conflict Resolution** folder. Template files open as new, untitled documents to which you can make changes and save without fear of accidentally changing the template file itself.

The following paragraphs describe how to set up a Cloud document from scratch. You can skip to the next section if you are using the template.

A Cloud is based on [Necessary Condition Thinking](#). Since Flying logic documents are set up for Sufficient Cause Thinking by default you will want to set the Operator popup menus as follows:

- Entity Operator: Fuzzy And (AND)
- Default Junctor Operator: Fuzzy Or (OR)

Clouds are read from left-to-right, starting at the Common Objective. However, this means the flow of the edges (arrows) must be towards the Common Objective or right-to-left: so you will want to set the Orientation popup menu to **Right to Left**.



Clouds are created using the entity classes in the built-in Conflict Resolution domain, and use the following classes: Want, Need, Common Ob-

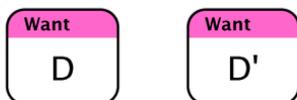
jective, Conflict, and Solution.

- ☑ Conflict Resolution
- Want
- Need
- Common Objective
- Conflict
- Solution

If you're using the template mentioned above, then the diagram is already drawn for you— you only need fill in the text. But the steps below assume you are drawing a cloud from scratch.

Step 1: Identify the Wants

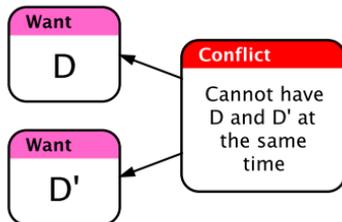
Create two **Wants** entities and give them titles that succinctly summarize each of the conflicting positions. Traditionally the two Wants are called **D** and **D'** ("D Prime").



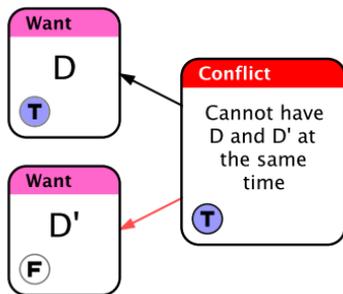
Step 2: Identify the Conflict

Create a single **Conflict** entity and make it a predecessor of each of the two Wants. If you're using the right-to-left orientation typical of Clouds, the Conflict entity will be to the right of the Wants.

Give the Conflict entity a title that summarizes *why* the Wants conflict.



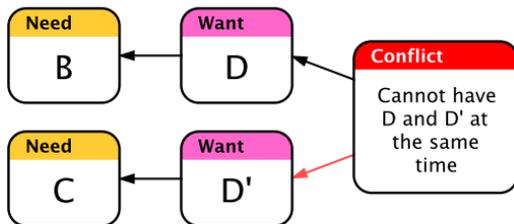
Finally, right-click (Mac or Windows) or control-click (Mac) one of the two edges leading from the Conflict entity and select **Negative** in the popup menu that appears. This causes the edge you negated to turn red. By doing this, our model accurately reflects the mutually-exclusive nature of the two wants. To see this, click the Show Confidence Spinners switch in the toolbar, then adjust the spinner on the Conflict entity from its maximum (True) to its minimum (False). You will see how the spinners on the Wants entities cannot both be true at the same time, due to the negated edge.



Step 3: Identify the Underlying Needs

Create two **Needs** entities, each one a successor to one of the Wants entities. The purpose of a position is to fulfill an underlying need. Give each need a title that summarizes the immediate need that its side in the conflict is trying to fulfill by asserting its position (the Want.)

The difference between a Need and a Want is simple: fulfillment of Needs are *conditions* considered necessary to fulfilling the overall objective, while Wants are particular *actions* chosen to fulfill the needs.

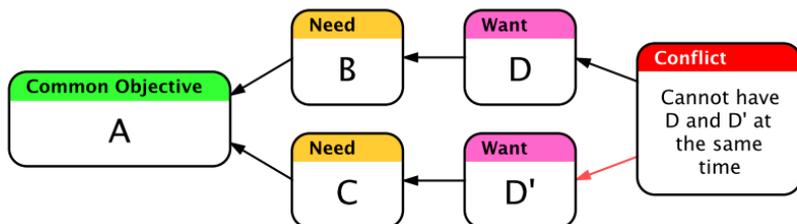


Step 4: Identify the Common Objective

Create a single **Common Objective** entity, and make it a successor of both needs. In a left-to-right orientation, the Common Objective will be the left-most entity in your diagram.

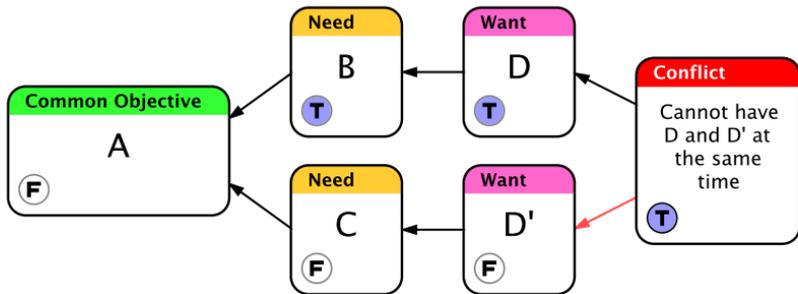
If the two sides in the conflict have no common objective, then there isn't really any conflict because the two sides could simply go their separate ways—they have no reason to cooperate. Thus, in every situation identified as a conflict, there is always a common objective. The Needs identified in the previous step are both considered *necessary* to achieving the common objective. In other words, both sides of the conflict believe that unless their needs are met, the common objective *cannot* be met.

The Common Objective is also usually at a “higher level” than the Wants or the Needs. In the case of the children fighting over a toy, the Common Objective might be for them to “get along and have fun.” Notice that this Common Objective doesn't mention the specific toy that is the subject of the conflict, even though the Wants and Needs may all mention it.



Step 5: Ensure Clarity by Reading the Diagram

Now that the diagram is complete, show the Confidence Spinners, and note that there is only one driver—the Conflict entity. This is the only entity that has no predecessors. If you have set up the document operators and negated one of the edges coming out of the Conflict entity as described above, you will see that by changing the value of the Conflict entity's spinner, one Want or the other can be satisfied (by becoming **True**), and yet the Common Objective can never become **True**. In other words, as long as the conflict exists, the Common Objective cannot be achieved.



Now read and revise the diagram for clarity and accuracy. Clouds are read from left to right, *against* the flow of the edges, using the pattern:

- “In order to **satisfy the need** we must **obtain our want**.”

This is the basic pattern of [Necessary Condition Thinking](#). This pattern applies to all the edges except the two coming from the Conflict entity. Once we have completed this step, we fix or clarify the wording.

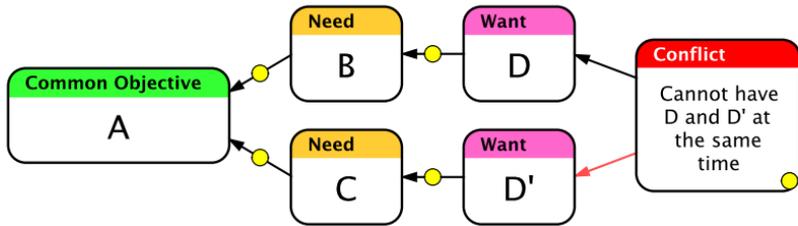
Step 6: Identify and Validate Assumptions

In the pattern from the previous step, there are two blanks for needs and wants. In this step we add a third blank:

- “In order to **satisfy the need** we must **obtain our want** because **of our assumptions**.”

Our assumptions are *why* we must obtain our want, and finding erroneous assumptions is the key to breaking the conflict. Assumptions “hide” underneath the Want→Need edges, and the Needs→Common Objective edges. There are also assumptions that underlie the Conflict entity itself— *why* we believe we can’t have both Wants simultaneously.

As you surface these assumptions, use Flying Logic’s annotation feature to add text to each of the four dependency edges and the Conflict entity. Take as much space as you need to describe each assumption, and begin each assumption with “...because”.



Sometimes there will be a single assumption under each edge, but often there will be several. Assumptions can be either *valid* or *invalid*. Invalid assumptions are often used to link needs to wants, but here you must critically evaluate each of the assumptions to determine their validity. Invalid assumptions can be eliminated, leaving just the valid ones.

Step 7: Propose Solutions

If we manage to invalidate *all* the assumptions on *any* of our edges, then we have eliminated the necessary condition relationship between two of the entities. If we have invalidated all the assumptions on the Conflict entity, then we have eliminated the perception of conflict itself. In either of these cases, the Cloud has “evaporated” and we have discovered there is, in reality, no conflict.

If the cloud is still intact, then we have at least one valid assumption in the five locations. The final step is to construct *solutions* (also called *injections*) that let us “break” one or more of the edges in the Cloud. A solution is an “option for mutual gain,” and the most constructive place in the cloud to find creative solutions is in the edges that connect the Wants to the Needs, by asking questions of this pattern:

- “How can we satisfy **Need** without obtaining **Want**?”
- “How can we accomplish **Common Objective** without satisfying **Need**?”
- “How can we obtain both **First Want** and **Second Want**?”

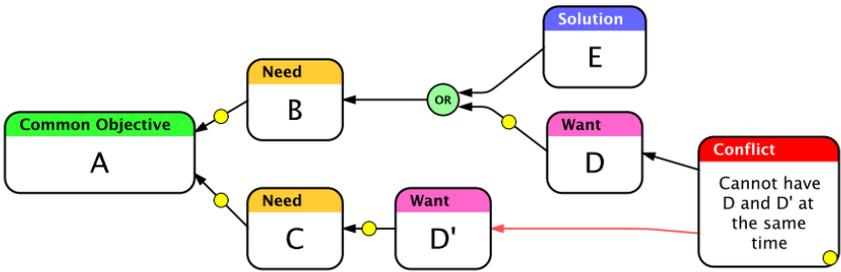
Remember that solutions that you come up with at this stage should not be considered final unless the situation you are analyzing is rather simple— this tool is for brainstorming a new set of options. You can use a [Future Reality Tree](#) to solidify the ideas you generate at this stage. Also, avoid the temptation to look for a single “panacea” solution— you will

often need two or more injections to implement a truly win-win solution.

To inject a solution into the diagram, first select the edge where you want the solution to appear then either:

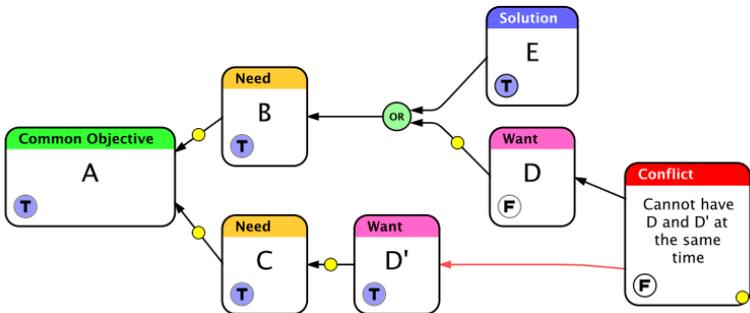
- Double-click the Solution entity in the class list in the sidebar.
- Right-click (Mac or Windows) or control-click (Mac) the edge and select **Solution** from the popup menu that appears.

An **OR** junctor will be inserted into the edge, and the new Solution entity will be added as a predecessor. Give the new Entity a title that summarizes the solution.



More solutions can be added to the same edge by selecting just the junctor, then using the same command from the popup menu.

By displaying the Confidence Spinners, you will see that you now have additional points of control for every solution you have added, and that it is now possible to make the Common Objective **True**, even if both Wants (the original positions) are not obtained.



Future Reality Tree

Perhaps you have a system you want to improve, and you've done a [Current Reality Tree](#) to identify *What needs to change*. Perhaps you've also done one or more [Clouds](#) to create some potential win-win solutions, in other words *What to change to*. But...

- How can you be confident which of those ideas will work?
- How do you pick one idea over another?
- How do you know something important hasn't been ignored or overlooked?
- What are the solution's strengths, and how can they be maximized?
- How can you be confident the "solution" won't have unanticipated effects that leave you in a situation that's worse than before?
- Are a potential solution's shortcomings something we can live with, something we can fix after the fact, or something we should avoid at all costs?

It is the job of the Future Reality Tree (FRT) to help you answer these questions.

The FRT is easiest-understood by contrasting it with the Current Reality Tree (CRT):

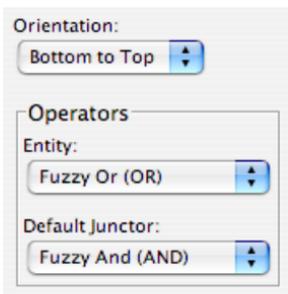
- To build a CRT, start with a set of Un-Desirable Effects (UDEs), and build down to the Core Driver, from which we invent Solutions (also called **injections**.)
- To build a FRT, start with a potential Solution (injection), and build upwards to a set of Desirable Effects (DEs).

FRTs can be built not only from a previously-conceived Solution, but also from other parts of previously-created CRTs and Clouds.

Flying Logic Setup

An FRT is based on [Sufficient Cause Thinking](#), and this is how Flying Logic documents are set up when first created, so you do not need to do anything special with the Operators popup menus to start creating your FRT. Most FRTs are drawn with one or more proposed Solutions at the bottom and the Desired Effects at the top, so you may want to use the

Orientation popup menu to change the orientation of your document to **Bottom to Top**.



FRTs are created using the entity classes in the built-in Effects-Based Planning domain, and primarily use the following classes: Desirable Effect, Un-Desirable Effect, Precondition, Intermediate Effect, and Action. If starting with solutions created from a Cloud, then FRTs will also often use the Solution class from the Conflict Resolution domain.

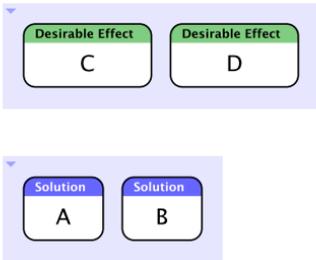
- | | |
|--|---|
|  • Effects-Based Planning |  • Conflict Resolution |
| •  Goal | •  Want |
| •  Intermediate Effect | •  Need |
| •  Precondition | •  Common Objective |
| •  Action | •  Conflict |
| •  Un-Desirable Effect | •  Solution |
| •  Desirable Effect | |

Step 1: State the Proposed Solution and Desired Effects

Create one or more Solution entities to identify the set of injections you plan to implement. These injections may come from a Cloud you've already created, and you can use the Copy and Paste commands to easily add these entities to your FRT document.

Also create one or more Desirable Effect entities that summarize the outcome you are working towards.

You may wish to temporarily group these two sets of entities to keep them separate— the purpose of the FRT is to fill in the “middle”.

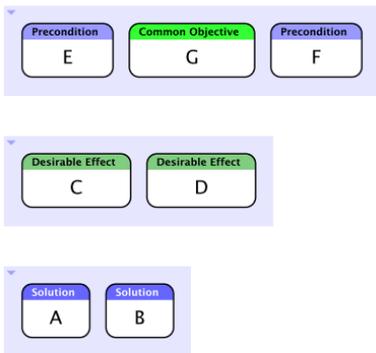


Step 2: Add Other Elements Already Developed

If you have already created a CRT, look for Precondition entities (statements about existing reality) that may be needed in your FRT. You can use the Copy and Paste commands to easily transfer them from your CRT to your FRT.

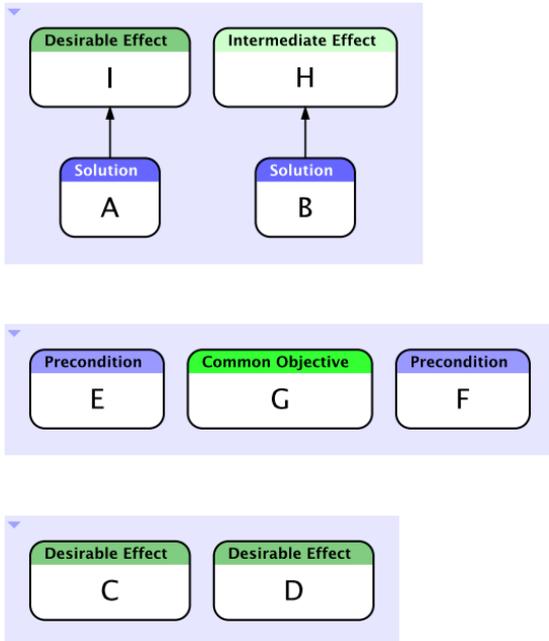
If you are working from an existing Cloud, also copy over the Common Objective and any of the Needs entities that the proposed Solutions are intended to satisfy.

You may wish to group the entities you’ve added in this step, as they represent entities that will end up in the “middle” of your FRT.

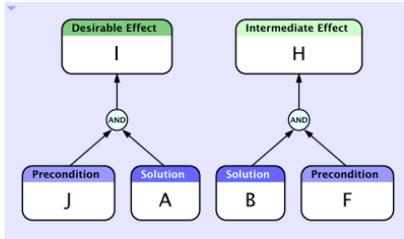


Step 3: Fill In the Gaps

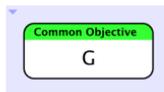
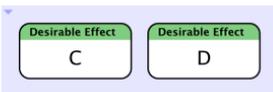
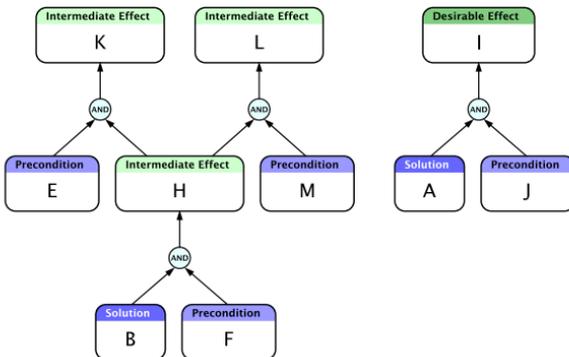
Starting with your Solution entities, add entities that represent the direct, inevitable consequences of those injections being put into place. Use Un-Desirable Effect entities for negative consequences, Desirable Effect entities for positive consequences, and Intermediate Effect entities for neutral consequences.



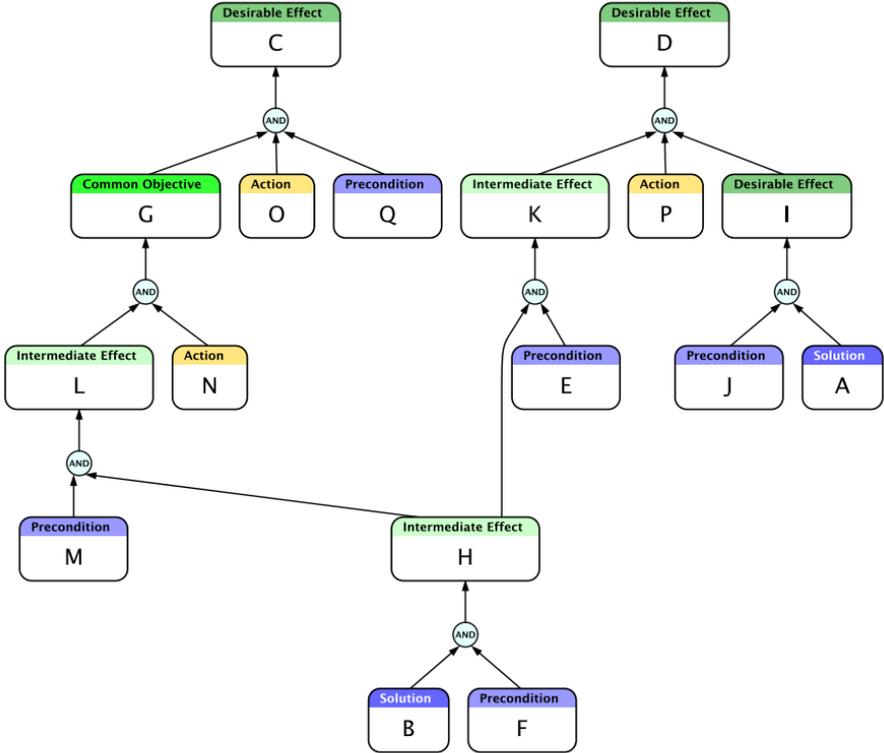
Use the [Categories of Legitimate Reservation](#), to check your causal connections. If the consequences you add are not sufficient by themselves, then make sure you add any Precondition entities, or tie in any other entities that express the other necessary conditions (AND relationships) needed to produce the predicted result. Feel free to move objects between groups or ungroup the entities when the edges begin giving your document structure.



Continue advancing from the effects you've identified to additional effects, evaluating whether the subsequent effects are bringing you closer to any of your Desirable Effects, or the Common Objective or Needs entities you may have added from your Cloud. If they do not, continue adding and evaluating effects you may not have previously considered.



If your progress slows down or stops, then consider additional Action entities you might add. These Actions are also injections, but to differentiate these injections from those that are part of your original solution, use the Action entity class instead of the Solution entity class you started with.



Step 4: Read and Verify the Tree

Once you have made connections to all of your Desirable Effects, re-read the entire diagram. Remember that FRTs are created using Sufficient Cause thinking, so the basic pattern you will use when reading is:

- If **cause A** then **effect B**.

When two or more arrows enter an entity, we have multiple *sufficient conditions*, also called an OR relationship:

- If **cause B** or **cause C** then **effect D**.

When two or more arrows enter an AND junctor, then we have multiple *necessary conditions*:

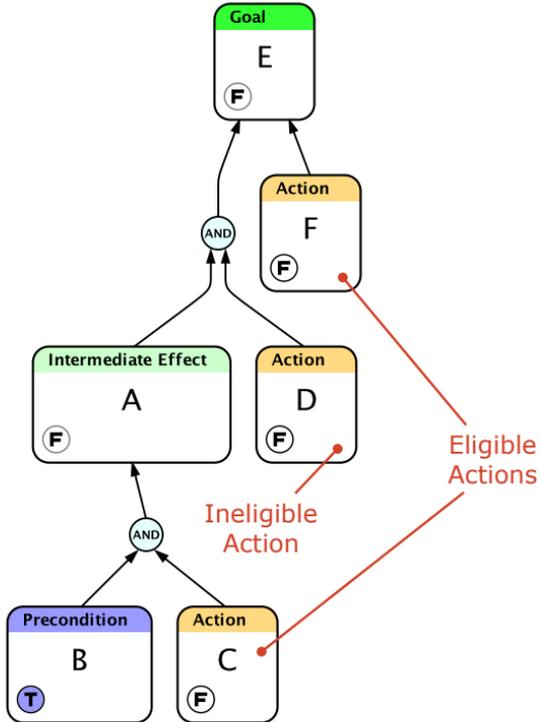
- If **cause E** and **cause F** then **effect G**.

Pay careful attention to the [Categories of Legitimate Reservation](#). Make sure every statement in your entities and those implied by the causal relationships are clear and logical.

When reading through the diagram, it is also a good idea to display the Confidence Spinners and use them as an aid to checking your logic.

1. Display the Confidence Spinners by clicking the **Confidence** button in the toolbar or selecting the **View ▾ Confidence** command
1. Set every spinner to **indeterminate** by using the **Entity ▾ Reset Confidence** command.
2. Because Preconditions are supposed to be facts about the world, set each Precondition entity's confidence value to **True**.
3. Notice that your Solution by itself is sufficient to cause additional effects, so set its confidence value to **True** and notice how those effects become true.

3. Notice that some of your Actions are “paired up” by AND junctors with other entities that are now **True**. These Actions are **eligible** for execution, while other Actions that are paired up with any entities that are not already **True** are **ineligible** for execution— they must wait until the other necessary conditions become **True**.

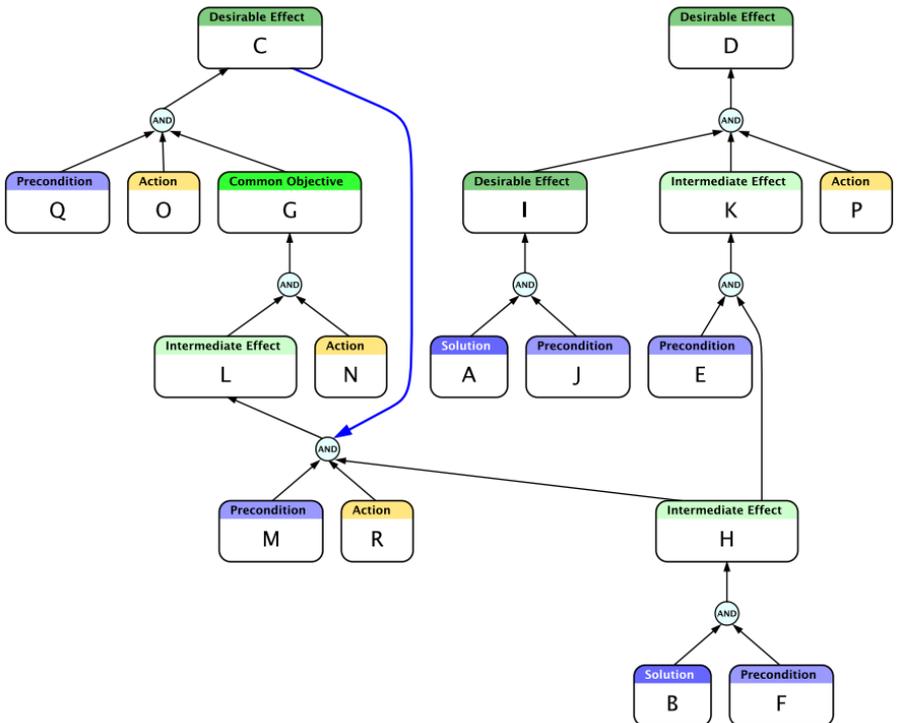


4. Continue step-by-step through the diagram, telling yourself the “story” of the diagram as you set each Action to **True** when it becomes eligible, until your Desirable Effects also become **True**. Correct any errors you discover in your logic along the way.

Step 5: Build In Positive Reinforcing Loops

Recall that when building a Current Reality Tree, occasionally there are Un-Desirable effects that are so severe that they “feed back” on others and create negative reinforcing loops. When creating a FRT, you want to loop for opportunities to do the opposite: build in *positive* reinforcing loops. If you can do so, you are more likely to create a solution that is self-sustaining.

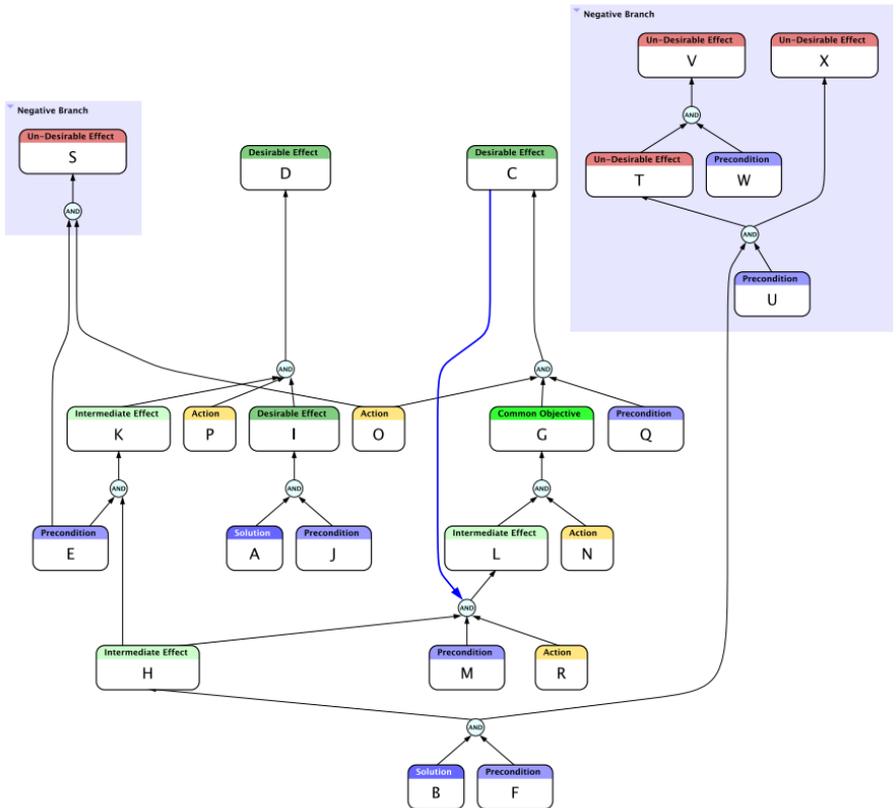
Look for Desirable Effects that may intensify effects lower in the tree that lead back to one or more Desirable Effects. If you find such cases, annotate them using [Back Edges](#). Pay close attention to where you might need to add additional Actions in order to create sufficient cause for a positive reinforcing loop’s existence.



Step 6: Seek and Address Negative Branches

This is a critical step. Whether or not you've already added some Un-Desirable Effects to your FRT, now is the time to go back over it and carefully search for other UDEs that are consequences of any of the entities we have added.

Once you have done that, look for the earliest places in the causal chain where UDEs start to appear. These "turning points" are the start of **Negative Branches**. It is critical that you deal with Negative Branches in order to avoid creating worse problems than those you set out to cure.



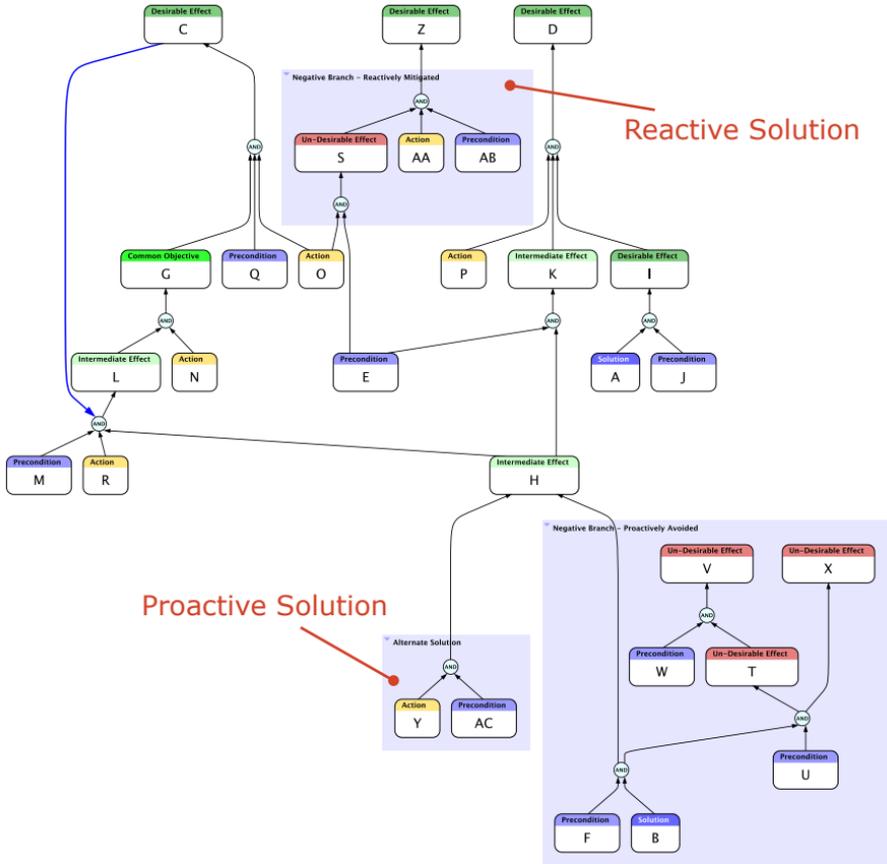
There are two approaches to addressing Negative Branches: **Reactive**

and **Proactive**.

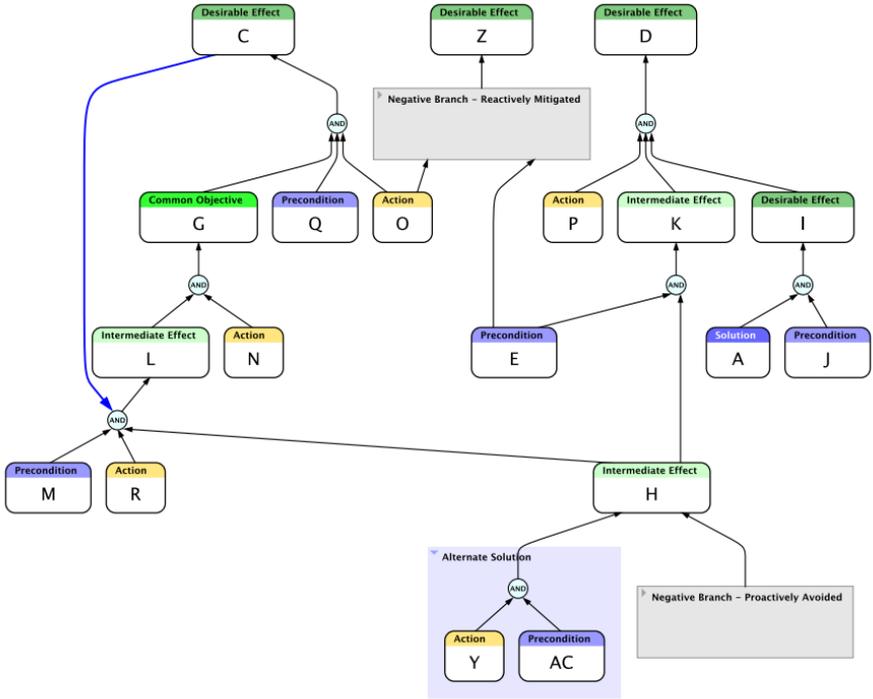
In the Reactive approach, UDEs are allowed (perhaps even expected) to occur, but are deemed unavoidable. New Action entities (injections) are then paired with the UDEs (along with other entities as necessary) to cause additional effects that mitigate the UDEs. These additional effects are hopefully Desirable Effects, but can also be neutral Intermediate Effects.

In the Proactive approach, alternative injections are created that achieve the next stage of Intermediate Effects that are on the path to the final Desirable Effects, *without* causing the UDEs. This is also known as “trimming the Negative Branch.”

In the illustration below, we deal with one Negative Branch *proactively* by discarding one of our original Solution entities **B** and devising an alternate course of action **Y**. The other negative branch is handled *reactively* by devising a new Action **AA** that mitigates the UDE **S** if and when it occurs.



Once a better path has been created, it is a good idea to keep a record of the entities that participated in the Negative Branch by keeping them in collapsed groups, instead of deleting them.



When someone brings you a well-intentioned proposal that you have concerns about, it is good practice to ask for some time to think about it, then take their proposal and construct a FRT with their suggestion as the initial injection at the root, and with the Desirable Effects predicted by the suggester *and* the UDEs you foresee as the final outcome. Once you have this FRT, you can discuss it in detail with the suggester. If you or they can develop injections that address the UDEs, then you are likely to have a proposal you can approve.

Prerequisite Tree

Perhaps you've gotten a picture of your Core Drivers using a Current Reality Tree (CRT). You may have used a Cloud to come up with some promising solutions and used a Future Reality Tree (FRT) to develop a solution you think will work. But unless your situation is quite simple, you're not done yet. One of the most overlooked aspects of planning lies in determining the things we *need* but *don't have yet*: these are the obstacles that lie in our path. And as we develop ways to overcome these obstacles, further obstacles will often become visible. The Prerequisite Tree (PRT) is a tool that helps us to identify and see beyond every obstacle, and make sure that every necessary activity is included in our plan.

Flying Logic Setup

A PRT is based on [Necessary Condition Thinking](#). Since Flying logic documents are set up for Sufficient Cause thinking by default you will want to set the Operator popup menus as follows:

- Entity Operator: Fuzzy And (AND)
- Default Junctor Operator: Fuzzy Or (OR)

PRTs are usually read from top-to-bottom, starting at the Objective(s). However, this means the flow of the edges (arrows) must be towards the Objective(s) or bottom-to-top: so you will want to set the Orientation popup menu to **Bottom to Top**.



PRTs are created using the entity classes in the built-in Prerequisite Tree domain, and use the following classes: Objective, Overcome, and Milestone.

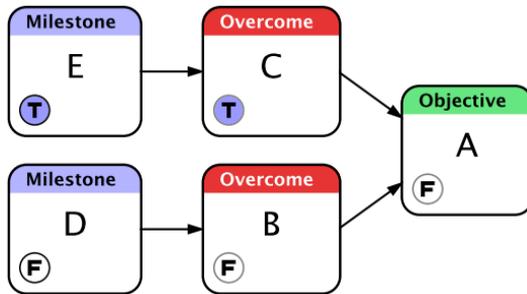
- Prerequisite Tree
- Objective
- Overcome
- Milestone

If you have already constructed PRTs in the past, the choice of **Overcome** instead of **Obstacle** as an entity class name may seem a little strange at first. We use the two terms somewhat interchangeably, but the name *Overcome* was chosen for three reasons:

- First, the choice of *Overcome* makes the tree more natural to read. For example, this simple sequence can be read, “In order to obtain **A**, it is necessary to overcome **B**. In order to overcome **B**, it is necessary to obtain **C**.”



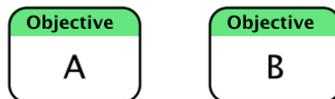
- Second, when using the Confidence Spinners to step through the logic of the tree, we use **True** to indicate that the statement in the title of the entity exists or otherwise pertains to reality, and **False** to indicate that it does not pertain. If we used the class name *Obstacle*, then a **True** confidence value would indicate the *existence of the obstacle*. However, what we want to express is the exact opposite: when all the necessary conditions are met, we want a confidence value of **True** to indicate that the obstacle no longer exists: *it has been overcome*. So when dealing with *Overcome* entities, it is easy to think of **False** meaning, “We have not yet overcome this,” (the obstacle exists) and **True** as meaning, “We have overcome this” (the obstacle no longer exists.) Thus, if every entity in our PRT does not have a Confidence of **True**, it is easy to see at a glance what we still need to accomplish.



- Third, there is a positive connotation to calling these entities *Overcome*. The name helps convey the idea that all obstacles contain the seeds of their downfall, and focuses the planner on the obstacle not as something that *exists to thwart them*, but rather as something that *exists to be thwarted*.

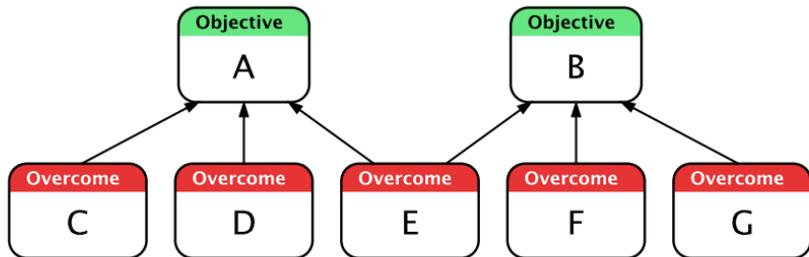
Step 1: Identify the Objective

Create an Objective entity and give it a title that uses simple, present-tense wording. Usually PRTs will have a single Objective entity that defines the outcome that you are working to achieve, although they can contain more than one Objective if they are closely related. Often the wording of the Objective will be drawn from an injection (Solution entity or Action entity) you used in creating a Future Reality Tree.



Step 2: Identify Some Obstacles to Overcome

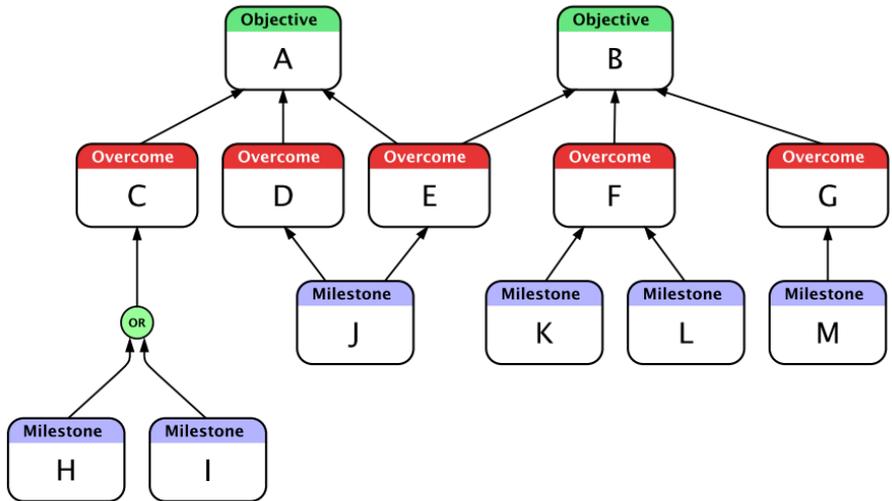
Something stands in the way of achieving your Objective, or you probably would have done it already. Create a set of Overcome entities that represent the *nonexistent* necessary conditions for achieving your Objective. The point here is not to list everything you will need to do to achieve your Objectives, but to identify *the things you still lack*. Connect each Overcome entity as a predecessor of your Objective.



Step 3: Brainstorm Milestones

Consider each of the Overcome entities you have added and brainstorm one or more Milestone entities that will negate the obstacles. It is useful to remember that you don't need to completely destroy an obstacle to get past it: you can (figuratively) go around it, over it, or under it—the point is to be creative.

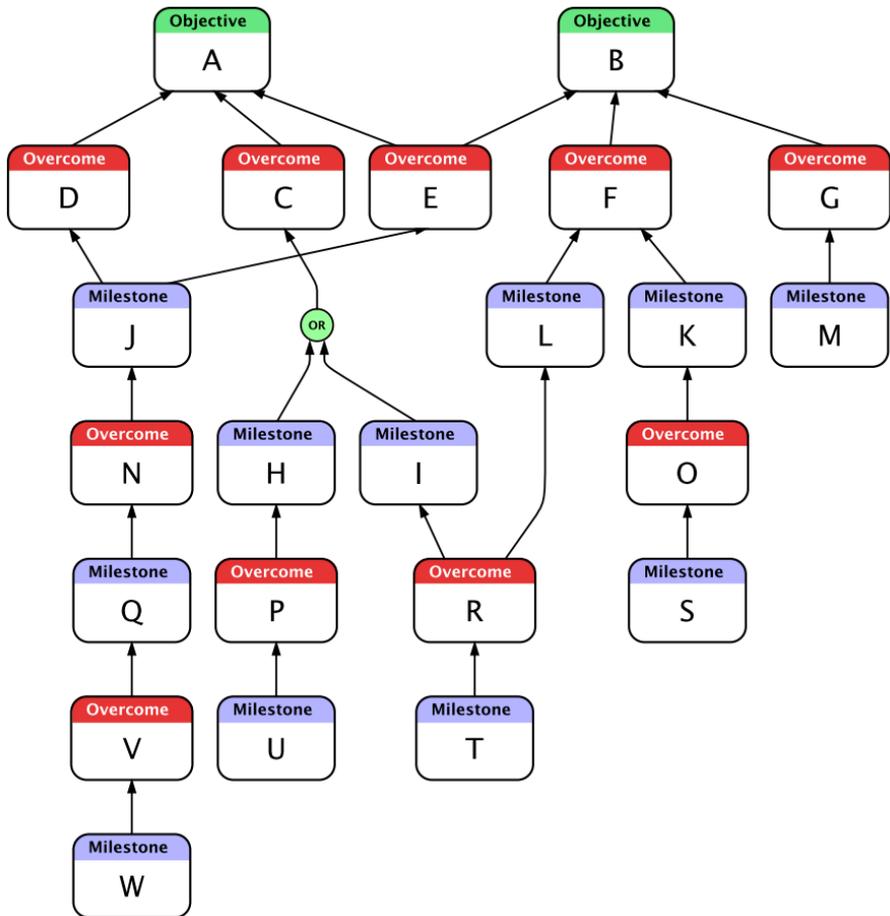
- Often there will be a single Milestone matched with each Overcome. (**M** is necessary to overcome **G**.)
- Some of the Milestones you identify may Overcome more than one obstacle. (**J** is necessary to overcome **D** and **E**.)
- Other times, your brainstorming will come up with two or more alternatives that may be able to Overcome an obstacle. You use OR junctors to model this. Junctors are easily created by dragging from an existing entity to a line. (Either **H** or **I** are necessary to overcome **C**.)
- And sometimes, more than one Milestone will need to be achieved in order to Overcome an obstacle. (**K** and **L** are necessary to overcome **F**.)



Step 4: Continue to Deepen the Tree

Now consider each of the Milestones you added in the last step. What obstacles to implementing them present themselves? Lack of knowledge? Lack of manpower? Lack of money? Creating a PRT is focused on finding those Necessary Conditions that you currently lack— this is all an obstacle really is. For each such obstacle you identify, create a new Overcome entity and connect it as a predecessor to the appropriate Milestone. For each of these new Overcome entities, devise Milestones that address them, and so on.

As you deepen the tree, the Milestones you add will start to have a smaller, more tractable character. At some point you will add Milestones for which you are unable to find any significant obstacles to their implementation. These Milestones are the roots of your PRT, and represent the accomplishments that must be tackled first. Of course, there may be many actual *Actions* that are required to implement a Milestone, and this is the subject of the Transition Tree discussed in the next chapter. But for now, it is sufficient to identify Milestones that entail no significant obstacles.



Step 5: Read and Verify the Tree

Once you feel that your tree is well-connected from its simplest Milestones all the way through to the ultimate Objective, it is time to carefully read the tree for clarity and completeness. Since PRTs are constructed using Necessary Condition thinking, the tree is read *against* the flow of the edges starting with the Objective. Each step of the tree is read with one of the following patterns:

- In order to obtain **A** it is necessary that we overcome **B**.
- In order to overcome **B** it is necessary that we obtain **C**.

Make sure that you apply the Categories of Legitimate Reservation as you read through the tree. Ask yourself questions such as:

- Do we really need to overcome this obstacle?
- Is there a way to avoid having to overcome this obstacle?
- Does the milestone really overcome the obstacle?
- Are there any other milestones required to overcome the obstacle?
- Are there any other milestones that are also sufficient to overcome the obstacle?

It is also a good idea to use Flying Logic's Confidence Spinners at this stage to go through your diagram step by step *with* the flow of the edges from the root Milestones all the way to the Objective.

Step 6: Trim and Finalize the Tree

Once you have verified that your PRT is logically sound, it may contain one or more alternate Milestones (connected by OR relationships) that you can now choose among. Trim the rejected alternatives either by deleting them or placing them within collapsed groups.

Transition Tree

Once you have identified the obstacles that stand in the way of achieving your goal and developed milestones that will overcome them, you need an execution plan: a set of actions that combine step-by-step to bring your system inexorably closer to its goal. Others who read your plan (and ideally participated in its creation) should be able to clearly see how every action, and particularly the actions in which they play a role contribute to the benefit of the entire system. This is the key to creating **buy in**— a shared vision that yields enthusiastic cooperation. The Transition Tree (TRT) is an effective tool for creating an execution plan that creates a transition from the *current reality* to a *future reality*.

Although a Transition Tree is related to the more traditional [PERT diagram](#) used in Project Management in that they both contain a set of sequenced actions, one of their main distinctions is the TRT's inclusion of Preconditions (assumptions about reality) paired with each action. This means that TRTs can contain numerous contingency plans that are triggered by the Preconditions that pertain at the time the plan is executed. Essentially, as execution of the plan proceeds, numerous different PERT charts can “fall out” of a TRT depending on what the situation “on the ground” looks like. This makes the TRT an ideal tool for creating plans that involve a significant degree of risk.

Flying Logic Setup

A TRT is based on [Sufficient Cause Thinking](#), and this is how Flying Logic documents are set up when first created, so you do not need to do anything special with the Operators popup menus to start creating your TRT. TRTs often flow upwards, with the Goal at the top. So you may want to use the Orientation popup menu to change the orientation of your document to **Bottom to Top**.

Orientation:
Bottom to Top

Operators

Entity:
Fuzzy Or (OR)

Default Junctor:
Fuzzy And (AND)

TRTs are created using the entity classes in the built-in Effects-Based Planning domain, and primarily use the following classes: Goal, Precondition, Intermediate Effect, and Action. You can also use Desirable Effect entities to highlight other positive benefits of your plan, and Un-Desirable Effect entities if your sequence of actions causes unavoidable UDEs that further part of the execution plan must address.

- Effects-Based Planning
 - Goal
 - Intermediate Effect
 - Precondition
 - Action
 - Un-Desirable Effect
 - Desirable Effect

Step 1: Identify the Goal

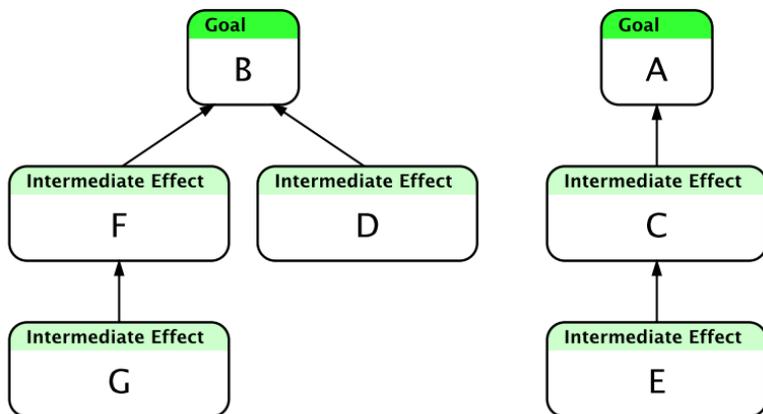
A TRT often contains a single Goal entity, but can contain more than one if they are reasonably related. You can start a TRT with an intuitive pre-conception of what your Goal should be, or you can start with your Goal taken from one of the injections taken from a Future Reality Tree, or an Objective taken from a Prerequisite Tree. In any case, the Goal entity should have as its title a clear, present-tense statement of the desired reality.

Step 2: Identify Intermediate Effects

If you have already done a Prerequisite Tree, you have a set of Milestone entities that you can copy directly into your Transition Tree document. It's important to realize, however, that while the Milestones in a PRT are

all *necessary*, they probably aren't sufficient. The PRT is used for identifying and overcoming *the things you don't have yet*, while the TRT is used for identifying *everything* you need to do, and the order in which you need to do them.

If you are not copying Milestone Entities from a PRT, you may want to create a number of Intermediate Effect entities that represent states you know will need to achieve along the way to your goal, and link them with edges to their order is more or less defined. It is not necessary to be absolutely rigorous at this stage— defining the exact causal sequence is the focus of the following steps.



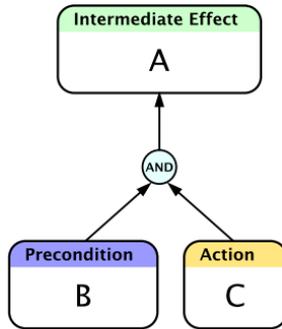
Step 3: Define a Complete Step

A *step* of your execution plan requires three things:

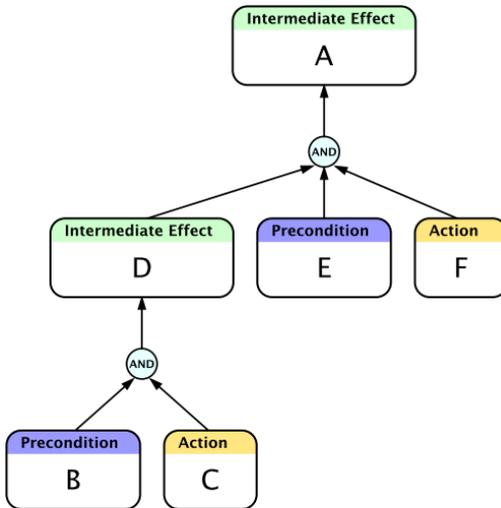
- The outcome you want to achieve. This is either an Intermediate Effect, a Milestone copied from a PRT, or the Goal of your TRT.
- A statement of current reality. This is either a Precondition entity, which represents an aspect of reality that is out of your control and which must therefore be taken as a given, or an Intermediate Effect or Milestone that was the outcome of a previous step.
- An Action. To be well-defined, an Action must be something within your control or influence, with clear criteria for determining that it has been carried out successfully, and must be something that can

be assigned to a resource with the responsibility and power to carry it out.

The current reality and action must logically combine as the necessary and sufficient causes of achieve the outcome.

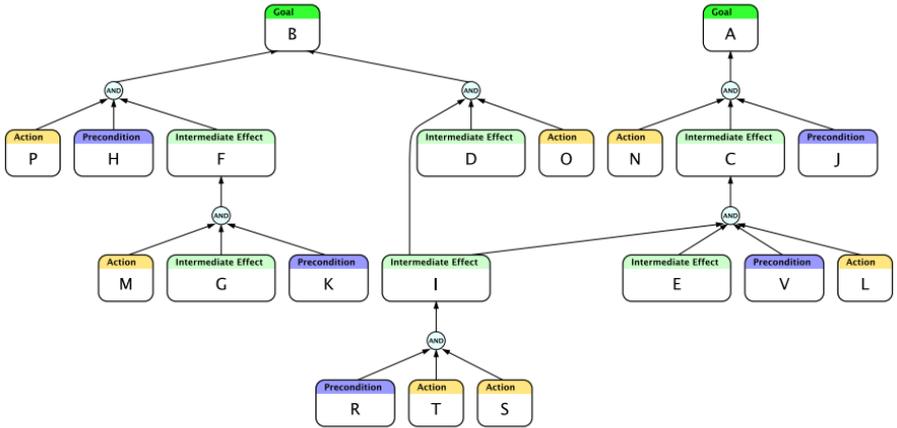


If in reading your step, the action is not sufficient to produce the outcome, then it needs to be broken down into one or more sufficient sub-steps.



Step 4: Continue Building the Tree

Each of the intermediate effects in your TRT must similarly be characterized as complete *steps*: outcome of actions and current realities. Often these steps will form a linear sequence, but other times they will diverge into parallel sequences, or have more complex dependencies.



Step 5: Seek and Address Negative Branches

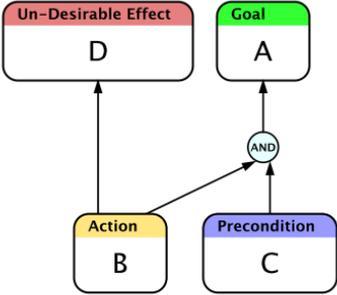
This is similar to the step of the same name in the description of the [Future Reality Tree \(FRT\)](#). In fact, a TRT can be thought of as a kind of FRT where instead of starting with an injection and ending up with the consequences, you start with a desired consequence (the Goal) and work backwards to the injections that will achieve it.

If you are working from a FRT you created previously, then your actions may already be designed to avoid the Un-Desirable Effects (UDEs) that are the hallmarks of negative branches.

On the other hand, sometimes it is impossible to avoid risk. Risk manifests as the failure of Preconditions (assumptions about reality) to be **True** when it comes time to execute the actions that depend on them. Depending on the nature of the environment in which the plan is executed, exactly *which* Preconditions may not hold true at the time the plan is executed can be very difficult to predict, and if you create a plan with a rigid picture of reality, you are likely to be disappointed when reality fails to conform. Thus, to the degree that your plan involves risks, it is

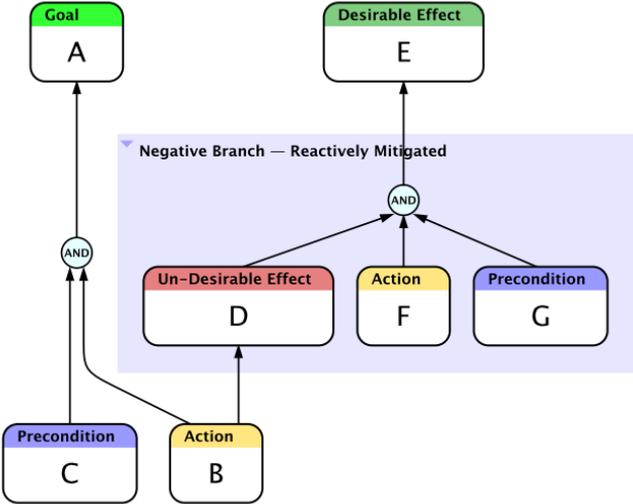
critically important that you identify the UDEs that can result from the failure of Preconditions, and develop alternative courses of action that either mitigate those UDEs (the *reactive* approach) or avert them (the *proactive* approach.)

If a plan terminates with any un-addressed UDEs, it is *incomplete*.

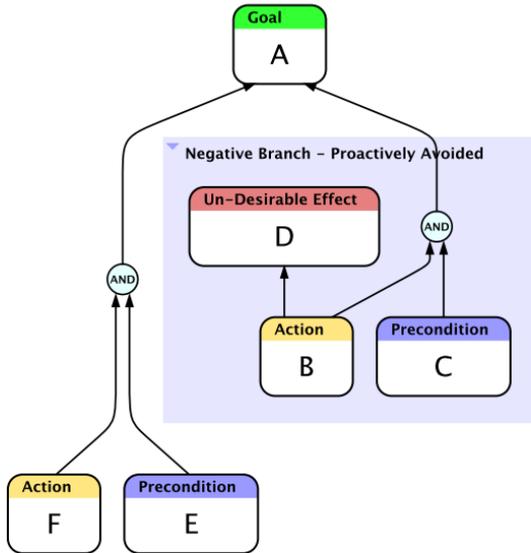


Incomplete Plan

A *complete* plan will terminate only with Goals or Desirable Effects.



Complete Plan – Reactive Mitigation



Complete Plan – Proactive Avoidance

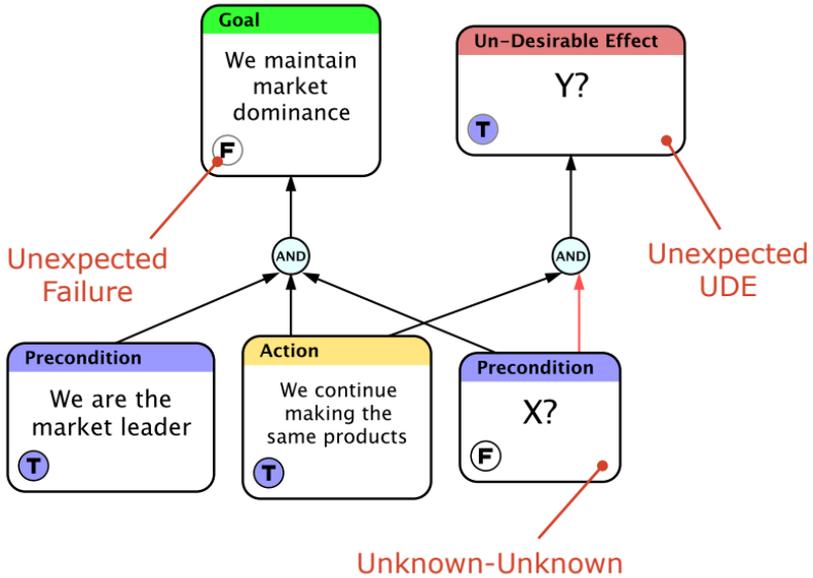
Finally, what happens if all the conditions determined to be necessary and sufficient for a particular effect are *present*, but when the plan is executed the effect turns out to be *absent*? What if some other UDE we didn't anticipate manifests? More importantly, how can we reduce the chances of this nightmare from occurring?

This is the case of **unforeseen uncertainty** (also called **unknown-unknowns**)— things that have not been and could not have been imagined or anticipated. In this case there can be no pre-determined contingency plan, but there are some things we can do to prepare:

- If possible, try several approaches in parallel and ultimately commit to the one that works the best.
- Avoid hubris: nurture an organizational culture of humility and resist being blinded by your own expertise.
- Be flexible and willing to adapt the plan to a changing situation.

- Give heed to hunches and concerns of experienced stakeholders, even if those reservations are not (for the moment) clearly articulated.

For the last case, even inarticulate reservations can be added to a TRT as unspecified Preconditions, and removed later if they fail to materialize. Don't add unknown-unknowns at every possible place— just where a strong-but-inspecific reservation has been expressed. Unknown-unknowns can also be added as part of assessment when planned effects fail to materialize as a plan is executed.



Step 6: Read and Verify the Tree

This is similar to the step of the same name in the description of the Future Reality Tree (FRT). If you have included contingency plans, then step through the pertinent parts of your tree more than once, each time setting up your Preconditions to trigger the different paths through your tree.

Strategy & Tactics Tree

The latest addition to the TOC-TP application tools, the Strategy and Tactics Tree (S&T Tree) is used to move from the highest-level organizational goals to a comprehensive, multi-tiered, fully-justified set of implementation steps. It is used to implement a wide-ranging improvement throughout a larger organization by making it clear what role every part of the organization plays.

Flying Logic Setup

An S&T Tree is based on [Necessary Condition Thinking](#). Since Flying logic documents are set up for Sufficient Cause thinking by default you will want to set the Operator popup menus as follows:

- Entity Operator: Fuzzy And (AND)
- Default Junctor Operator: Fuzzy Or (OR)

S&T Trees are usually read from top-to-bottom, starting at the highest-level Strategy. However, this means the flow of the edges (arrows) must be towards the highest-level Strategy or bottom-to-top: so you will want to set the Orientation popup menu to **Bottom to Top**.



S&T Trees are created using the entity classes in the provided domain file **Strategy & Tactics Tree.logic-d** in the **Examples/Strategy & Tactics Tree** folder. You can either import this domain into an existing document with the **Entity** \blacktriangleright **Import Domain** command, or open it with the **File** \blacktriangleright **Open** command, in which case it acts like a template document and opens a new, untitled document with the S&T Tree domain already imported and ready for use.

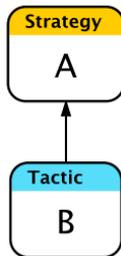
- ▣ Strategy & Tactics Tree
- ▣ Strategy
- ▣ Tactic
- ▣ Parallel
- ▣ Necessary
- ▣ Sufficient

Structure of the S&T Tree

The S&T Tree is based on the idea that **Strategy** and **Tactics** are complementary concepts used to describe a tree-like hierarchy of action, with each *Step* (node) of the tree justifying its existence with a strategy: a description of *why* the node exists. The highest-level strategy corresponds to the system's goal.

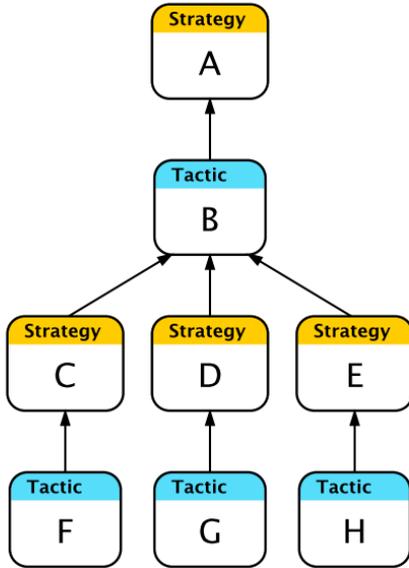


Each Strategy is supported by a single Tactic entity that describes *how* the strategy will be implemented. The bottom of a complete S&T tree will always be a layer of Tactics—the most fundamental actions that support the strategies.

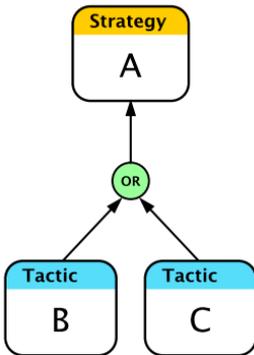


If more than one Tactic is necessary to implement a Strategy, a Tactic may be broken down into two or more sub-Tactics, but each one must first be justified by its own Strategy. Therefore, each Strategy always

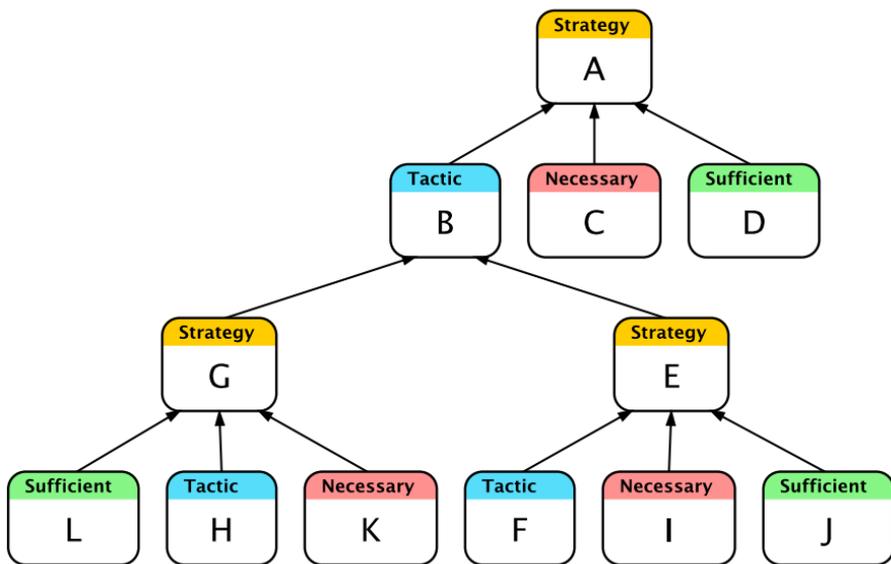
has *exactly one* Tactic below it, but tactics may have either zero, or two or more sub-Strategies.



If a Strategy has more than one possible Tactic that can accomplish it, then this can be added as an OR relationship.



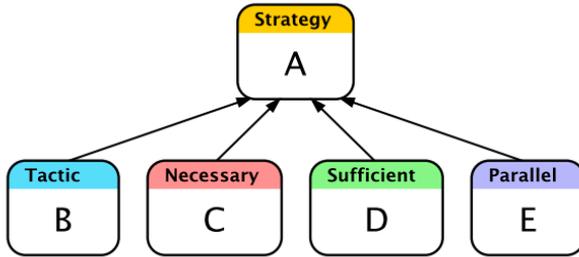
For a given Strategy, we need to do more than provide a Tactic for accomplishing it— we also need to justify that Tactic as both *necessary* and *sufficient* to accomplish its parent strategy. So we create a **Necessary** entity and a **Sufficient** entity and make each one a sibling of each Tactic entity. The title given to each entity should do exactly as the class name suggests: describe why the Tactic *must* be implemented to accomplish the strategy (Necessary), along with why that Tactic absolutely *will work* (Sufficient). If there are numerous justifications for why a Tactic is Necessary or Sufficient, then additional Necessary or Sufficient entities can be added, or they can be enumerated in the entity’s textual annotations.



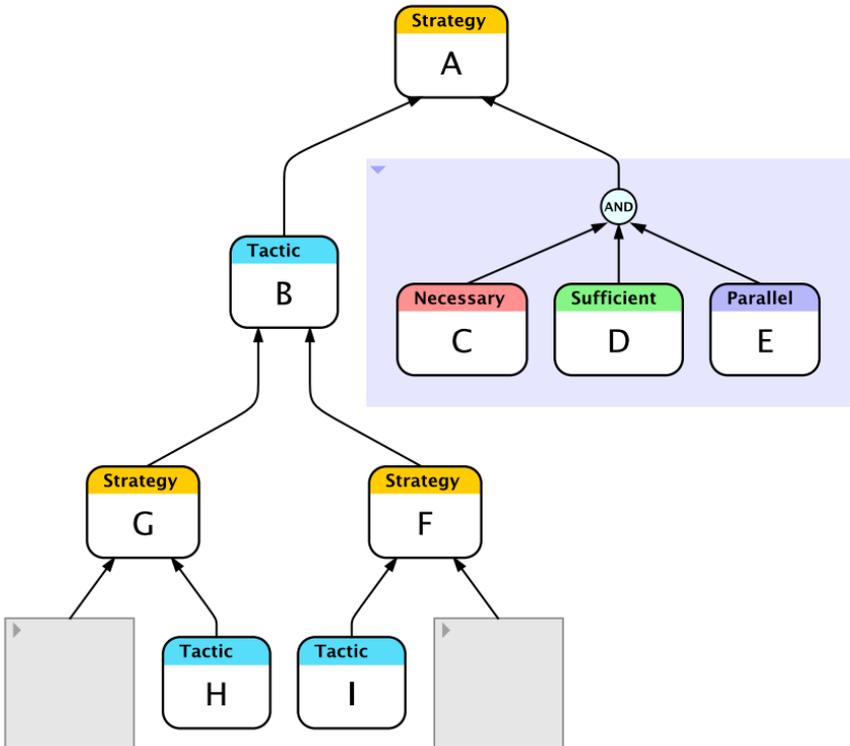
One more entity class, the **Parallel** (“parallel assumption”) class is used to proactively answer objections that neither directly address the Necessity or Sufficiency of the Tactic, such as:

- The Strategy already exists— no action need be taken to implement it.
- It is not possible to implement the Tactic.

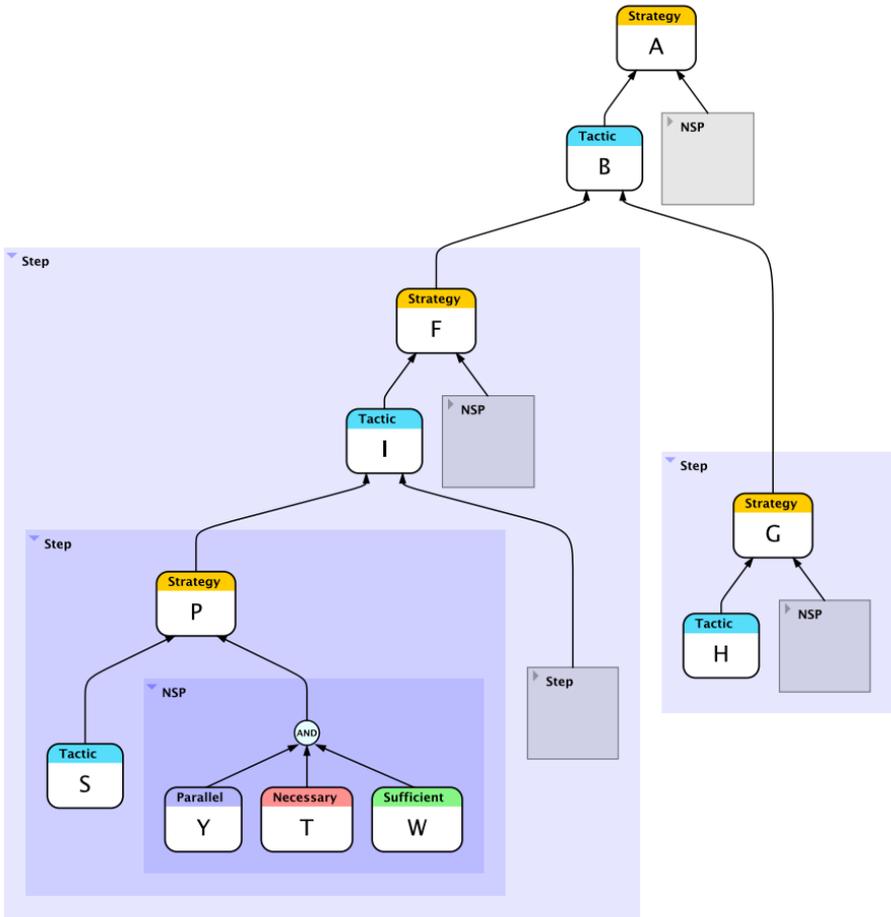
Taken together, all five kinds of entities constitute a **Step**.



Since S&T Trees can grow quite large, it is useful to use Flying Logic's grouping feature to manage the diagram. One approach is to group all a Strategy's supporting entities and use a junctor to combine their edges with an AND junctor so only a single edge emerges from the group.



Groups can, in turn, be used to group entire Steps, including the Strategy entity, the Tactic entity, and the sub-group containing the Necessary, Sufficient, and Parallel (NSP) entities. Using this technique, you can arrange a very large S&T Tree to make it easy to “drill down” to the level of detail you need. Take these ideas as suggestions, and feel free to develop your own techniques for managing large Flying Logic diagrams.



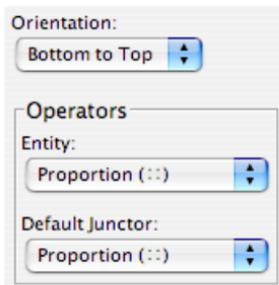
Part III — Other Techniques

Evidence-Based Analysis

This category of entity classes is suited to an environment when a more probabilistic mode of analysis is desired. One real-world scenario where Evidence-Based Analysis is useful is in [Competitor Analysis](#). Usually an analysis is designed and then carried out over a period of time. During such time, Propositions may be discovered to hold, which may trigger further actions by the agency conducting the analysis.

Flying Logic Setup

There are two styles of Evidence-Based Analysis: **belief-network** and **probabilistic**. If the belief-network style is used, the Flying Logic document is set up with Proportional ($::$) for both the entity operator and default junctor operator. If the probabilistic style is chosen, the document is typically set up with Sum Probabilities (\oplus) as the entity operator and Product (\times) as the default junctor operator. This setup is analogous to the use of Fuzzy Or (OR) and Fuzzy And (AND) in [Sufficient Cause Thinking](#).



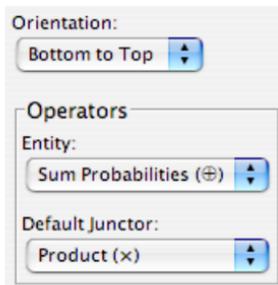
Orientation: Bottom to Top

Operators

Entity: Proportion ($::$)

Default Junctor: Proportion ($::$)

Setup for Belief Network



Orientation: Bottom to Top

Operators

Entity: Sum Probabilities (\oplus)

Default Junctor: Product (\times)

Setup for Probabilistic

Proposition

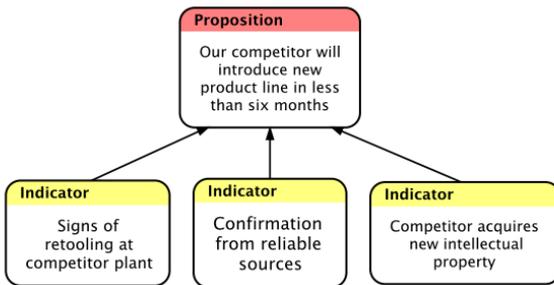
Propositions (also known as *requirements*) are questions for which the analysis is intended to discover the most likely answers. Propositions take the form of a statement that has some probability of being true. Determining whether the probability of the Propositions exceeds determined thresholds is a primary purpose of Evidence-Based Analysis. Propositions are analogous to goals in Effects-Based Planning, and thus are *terminal*, i.e., they are always successors and never predecessors.

Proposition

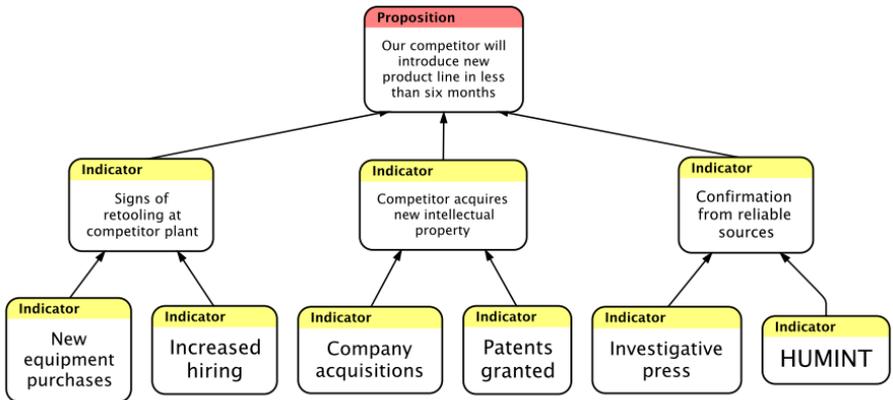
Our competition will introduce new product line in less than six months

Indicator

Indicators are potential causes for Propositions or other Indicators, and can be considered analogous to Intermediate Effects in Effects-Based Planning. Another way of thinking of Indicators is as inferred evidence. Each Proposition typically has a set of Indicators that feed into it, each of which is considered to be a possible cause of the Proposition, and which together form a “template” for recognizing that the Proposition holds (i.e., that the *requirement has been met.*)



Each indicator in turn may have a set of more specific indicators which feed into it and form a “sub-template” for recognizing that the indicator in question probably holds. Indicators are usually both successors and predecessors.

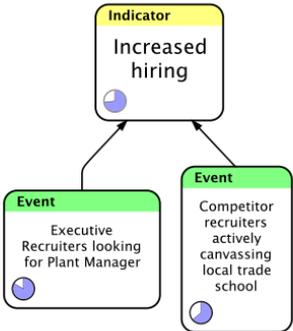


In a complex analysis, individual analysts can be assigned responsibility for certain indicators, which places them in a supervisory role over all indicators that are predecessors of the indicators for which they are directly responsible.

Event

Events represent direct evidence which becomes known throughout the life cycle of the analysis. In the intelligence community for example, Events may be derived from Signals Intelligence ([SIGINT](#)), Human Intelligence ([HUMINT](#)), or Open Source Intelligence ([OSINT](#)).

Events are always predecessors and are never successors. They are assigned a Confidence value based on their *reliability* (or probability.)



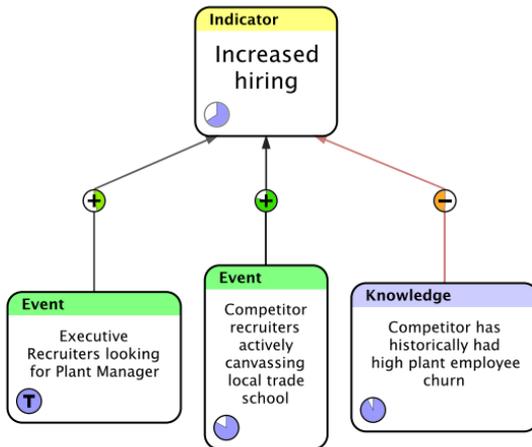
Knowledge

Knowledge represents pertinent facts known to be true about the situation under analysis. Knowledge can be built into the analysis before events are received, or can be added to the analysis in response to events as they occur. Knowledge entities are combined with Events to provide context and semantics either supporting or refuting the various indicators into which they feed.

Like Events, Knowledge entities are predecessors and not successors. They are assigned Confidence values based on their reliability (or probability.)

Edge Weights

Edge weights in the model are assigned based on the positive (or negative) *correlation* between each entity and its successors.

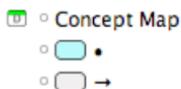


Concept Maps

[Concept Maps](#) are used to visualize and capture or convey a quick understanding of a web of related concepts.

Flying Logic Setup

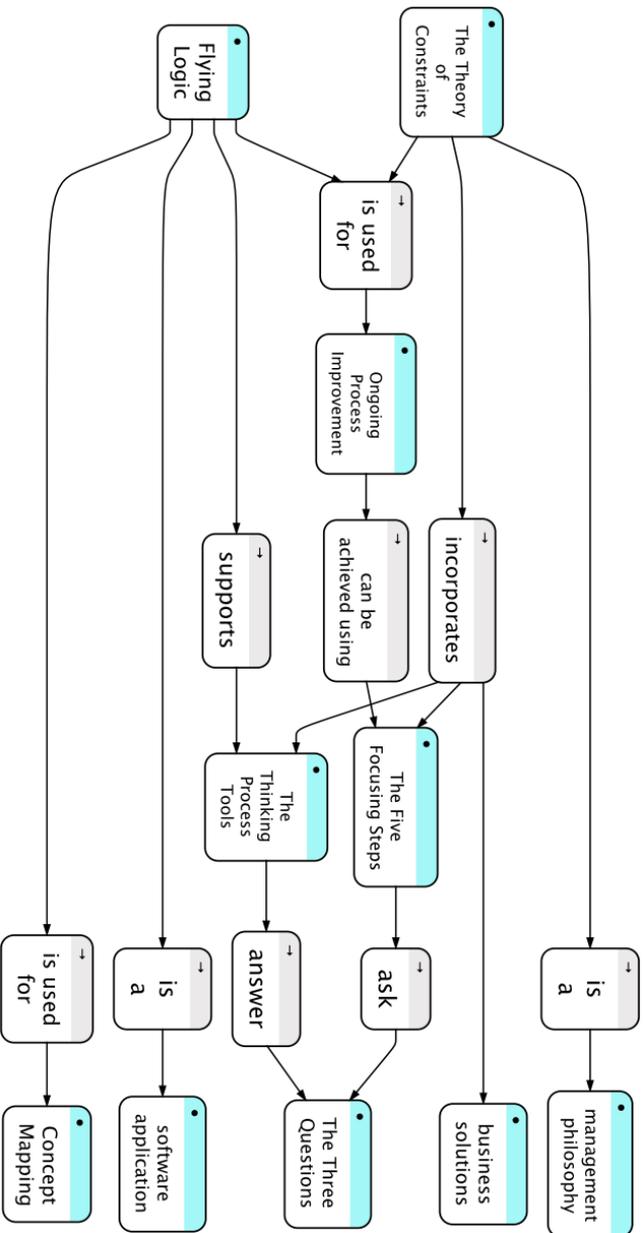
Concept Maps are created using the entity classes in the provided domain file **Concept Map.logic-d** in the **Examples/Concept Maps** folder. You can either import this domain into an existing document with the **Entity** → **Import Domain** command, or open it with the **File** → **Open** command, in which case it acts like a template document and opens a new, untitled document with the Concept Map domain already imported and ready for use.



Structure of Concept Maps

Concept maps use two entity classes, **Concept** (•) and **Relation** (→). Symbols were used for the entity class names instead of words because Concept Maps are read entirely from their entity titles, and the words “Concept” and “Relation” are never spoken.

Concepts Maps start with one or more main concepts at the root, and relations are used between concepts to connect in supporting concepts. The main rule when building Concepts Maps is that each Concept→Relation→Concept step should be readable as a complete sentence. Additional relevant concepts can be added in any order, and connected in as many places as they are used.



Appendix

Resources

Flying Logic Web Site

The resources at FlyingLogic.com are intended to be of interest not only to Flying Logic users, but also to people generally interested in TOC, business improvement, and personal improvement.

- [Flying Logic Forum](#) — Discussion
- [Flying Logic Wiki](#) — Collaborative knowledge base
- [Flying Logic Blog](#) — News and items of interest

Web Sites on the TOC

- [My Saga to Improve Production](#)
By Eli Goldratt
- [TOC.tv](#)
Videos by Eli Goldratt
- [A Guide to Implementing the Theory of Constraints](#)
Kelvyn Youngman
- [TOC Video Overviews](#)
Dr. James R. Holt, Washington State University
- [Theory of Constraints International Certification Organization](#) — Among other things, the TOC-ICO hosts a yearly conference
- [TOC Glossary](#)
Pinnacle Strategies
- [Strategy and Tactics](#) — a description of the S&T Tree
By Eli Goldratt, Rami Goldratt, and Eli Abramov
- [Throughput Accounting](#) — includes a description of Throughput, Inventory, and Operating Expense
Wikipedia

Books on the TOC

- [The Logical Thinking Process: A Systems Approach to Complex Problem Solving](#)
by H. William Dettmer
- [Thinking for a Change: Putting the TOC Thinking Processes to Use](#)
by Lisa J. Scheinkopf
- [Introduction to the Theory of Constraints \(TOC\) Management System](#)
by Thomas B. McMullen, Jr.

Books on Psychology, Communication, and Negotiation

- [Mistakes Were Made \(But Not by Me\): Why We Justify Foolish Beliefs, Bad Decisions, and Hurtful Acts](#)
by Carol Tavris, Elliot Aronson
- [Getting to Yes: Negotiating Agreement Without Giving In](#)
by Roger Fisher, Bruce M. Patton, William L. Ury
- [Getting Past No: Negotiating in Difficult Situations](#)
by William L. Ury
- [Difficult Conversations: How to Discuss what Matters Most](#)
by Douglas Stone, Bruce Patton, Sheila Heen, Roger Fisher
- [Crucial Conversations: Tools for Talking When Stakes are High](#)
by Kerry Patterson, Stephen Covey et al.

Other Useful Web Sites

- [The Theory Underlying Concept Maps and How To Construct Them](#)
by Joseph D. Novak, Alberto J. Cañas, Florida Institute for Human and Machine Cognition