2016 ■ Vol. 12 ■ no. 1

DOI: 10.20982/tqmp.12.1.p039

Systems Factorial Technology Explained to Humans

Bradley Harding[a,✉], Marc-André Goulet[a], Stéphanie Jolin[a], Christophe Tremblay[a], Simon-Pierre Villeneuve[a] & Guillaume Durand[a]

[a]School of Psychology, Universtity of Ottawa

Abstract ■ The study of mental processes is at the forefront of research in cognitive psychology. However, the ability to identify the architectures responsible for specific behaviors is often quite difficult. To alleviate this difficulty, recent progress in mathematical psychology has brought forth Systems Factorial Technology (SFT; Townsend and Nozawa, 1995). Encompassing a series of analyses, SFT can diagnose and discriminate between five types of information processing architectures that possibly underlie a mental process. Despite the fact that SFT has led to new discoveries in cognitive psychology, the methodology itself remains far from intuitive to newcomers. This article therefore seeks to provide readers with a simple tutorial and a rudimentary introduction to SFT. This tutorial aims to encourage newcomers to read more about SFT and also to add it to their repertoire of analyses.

Keywords ■ SFT, Cognitive processes, Information processing, architectures, Interaction Contrasts, Capacity measures

✉ bhard024@uottawa.ca

Introduction

A variety of cognitive processes allow us to treat our environment seamlessly and efficiently. Nevertheless, these processes are hardly accessible to researchers and are not without mystery. Because of the black box nature of the brain, questions regarding information processing and decision making are often left unanswered. While some research regarding a specific mental process may show evidence in favor of a particular architecture (the way information is processed; processing architectures are explored in detail in Section 2), said evidence remains mostly indirect. For example, researchers studying visual search paradigms have shown evidence in favor of both serial self-terminating processing (Treisman & Gelade, 1980) and parallel exhaustive processing (Cousineau & Shiffrin, 2004, e.g.), two completely different processing architectures. This difficulty to pinpoint and associate a specific architecture to a cognitive process has been present since the beginning of cognitive research, and a valid and complete analytical approach was needed to resolve this issue.

If the brain is a factory, producing thoughts and actions instead of products, Systems Factorial Technology (SFT; Townsend and Nozawa, 1995) is a probe examining how workstations are disposed. Expanding on Sternberg's additive method (Sternberg, 1969, 1998) and Donders's subtractive method (Donders, 1969), Townsend and Nozawa (1995) SFT methodology is a monumental addition to cognitive science that was brought forth to identify the underlying architecture of an information processing paradigm. To do so, the stimuli are manipulated and presented to participants in a Double Factorial Paradigm (which will be defined later in the text). Following these manipulations, participant response times (RT) are collected and their distributions subsequently transformed into a specific curve. It is this curve that is used to diagnose the processing architecture of the cognitive process. While SFT may appear as particularly focalized, it has been utilized in a variety of paradigms, such as visual search (Townsend & Nozawa, 1995; Fific, Townsed, & Eidels, 2008), local-global information processing for people affected by autism (Johnson, Blaha, Houpt, & Townsend, 2010), identification of different strategies for different classification task variants (Fific, Nosofsky, & Townsend, 2008), amongst others.

Despite its efficacy at identifying the processing architecture within the brain, this methodology and its applications can seem complex for newcomers. Although some research briefly instructs users on how to use SFT (Gaissmaier, Fific, & Rieskamp, 2011; Townsend, Fific, & Neufeld, 2007), to the best of our knowledge, there are no simple SFT tutorials for researchers to carry out the methodology with their own data. Therefore, this tutorial aims to simplify the SFT methodology and its various components by taking a step back from the technical aspects usually found in SFT articles and present a parsimonious step-by-step approach. The tutorial is divided in four sections. The first section elaborates an analogy between the brain and a high-tech, super-secret car plant. The second section gives the nomenclature and basic principles that any SFT user must learn and get accustomed to. The third section out-lines the SFT methodology and how to interpret its results. Finally, the last section briefly introduces the notion of capacity within an SFT analysis, a slightly more complex concept that gives users additional diagnostic power. In addition, we provide in Appendix A and B the steps to perform an SFT analysis and how to simulate RTs that correspond to all five architectures. These simulations will be presented using the Mathematica programming interface.

While reading this article, we encourage readers, especially newcomers to cognitive processing, to consult the *SFT Cheat Sheet* presented in Table 1. This table can also be used as a reference when consulting other SFT articles.

The nuts and bolts of SFT

Before hastily going deeper into the SFT methodology, consider an analogous situation to our brain: a car assembly plant. This plant is new, high-tech, under industrial secrecy, well-guarded, has unavailable blueprints, and it is impossible to neither directly observe the workstations, nor to question its employees. In short, it is incredibly difficult to gather any information on how the vehicles are built. The only freely available information is the moment when a chassis enters the plant and the moment the same chassis exits the plant as a full car. Suppose that two workers, Alice and Bob, are in charge of putting on the front wheel(s) of a car and that we wish to know their working stations' order.

Given the limited available information, the only way to find out the plant's setup is to indirectly influence Alice and/or Bob's productivity. For example, we could tamper with their mental or physical state by keeping either one or both of them up all night. Assuming that a tired employee works at a slower pace than a rested worker, the plant's overall production is affected depending on the workers' state of mind. Four plant operation-states are therefore possible: one where both workers are well-rested, one where both workers are tired, and two where one worker is well rested and the other is tired. These four working conditions are akin to those needed in an SFT analysis

Nomenclature and Basic Principles

Cognition can be viewed as a series of processes which can themselves be divided into many individual and specialized sub-processes. For instance, in a visual search task (the cognitive process), one sub-process could detect the stimulus' shape, another one could identify the color, a third one could detect movements, etc. When a sub-process' task is completed, it fires a signal and the moment at which this occurs represents the sub-process' overall completion time, RT.

Sub-processes can be paired and organized so that they either process information simultaneously or sequentially (processing order). The process they're a part of is completed when one or both sub-processes are ready to fire (stopping-rule). The combination of processing order and stopping rule creates the information processing architectures (Townsend & Ashby, 1983), which are precisely what SFT aims to detect. While it is possible to combine more than two sub-processes in a cognitive system (see Yang, Fific, Townsend, 2013 for SFT applied to n number of sub-processes), the current tutorial is limited to the standard SFT methodology that detects the architecture between pairs of sub-processes.

SFT can discriminate between the five following information processing architectures.

1. Serial self-terminating: sub-processes are queued sequentially, but only one needs to be completed for the process to fire. The order of sub-processes is unknown and wholly random. In the car-plant, an analogous situation would be the installation of the front wheel of a three-wheeled car; either Alice or Bob can install it, and once installed, the car moves to the next workstation.
2. Serial exhaustive: sub-processes are queued sequentially and the process fires when both sub-processes have finished. The order of sub-processes is unknown and SFT is unable to identify which one acted first. In the car-plant analogy, it is a scenario where one worker attaches their wheel before the other and the car only moves on once both wheels have been attached.
3. Parallel self-terminating: both sub-processes work simultaneously and the process fires as soon as one sub-process finishes. In the car-plant analogy, this could represent a scenario where the car moves to the next workstation as soon as one wheel is affixed (regardless of which worker completed the task). Of course, this scenario would result in poorly constructed automobiles in the car-plant analogy.
4. Parallel exhaustive: both sub-processes work simultaneously and the process fires only after both sub-processes have finished. Therefore, the sub-process with the slowest RT triggers the decision with the fastest sub-process waiting idly for it to finish. In the car-plant analogy, Alice and Bob work their own side of the car. They begin working at the same time and the car leaves when both wheels have been installed.
5. Coactive: both sub-processes work simultaneously and collaborate towards achieving a common goal. The sub-processes are broken into k sub-tasks that must all be completed for the main process to fire. These sub-tasks are shared amongst both sub-processes. In the analogy, we can break the installation of the front wheel of a three-wheeled car into screwing 10 lug nuts. It is possible that each worker screws in 5 apiece, but if, for any reason, Bob does poorly on a certain car, Alice can compensate for him by screwing more than half of the

lug nuts herself. Coactive is thus a form of collaborative processing (Miller, 1982).

Figure 1 is a visual representation of the five processing architectures. The sub-process's completion is represented by the circle and blue asymptote. The green arrow represents the moment at which the main process can fire, the black arrow represents the input coming in, and the grey arrow represents the transfer from one sub-process to another.

The RT for a cognitive process follows a distribution whose shape is different for each process. For example, in the redundant target attribute task, in which participants need to detect certain target dimensions of a stimulus, RT distributions usually resemble a positively-skewed Weibull distribution (Engmann & Cousineau, 2013). These RT distributions can be affected. Consider that an unimpaired sub-process treating optimal stimulus constitutes the High condition (or H) and that the same process working on a stimulus that is harder to discern constitutes the Low condition (or L). When the sub-process is in the L condition, the completion time should be slower than in the H condition.

This simple reasoning stems from SFT's core assumption, selective influence (Townsend & Nozawa, 1995). This assumption posits that it is possible to affect, or to selectively influence but a single process within an endless array of connections (Sternberg, 1969). It is therefore primordial in an SFT analysis to tailor stimuli so that it only affects the targeted process and sub-processes within. In addition, even if only a single sub-process is targeted, the factor's effect must be sufficiently large to yield an observable effect, that is, an effect that cannot be explained by other random factors alone. In short, the factor must sufficiently affect the processing capability of a single sub-process without affecting the other sub-processes (Dzhafarov & Schweickert, 1995).

Additionally, SFT posits that sub-processes must, in most cases, be stochastically independent for correct diagnoses (Townsend & Nozawa, 1995). This means that the sub-processes' processing times are not correlated; a correlation between sub-processes could result in model mimicking (where completely different architectures show similar results making it impossible to discern the architectures from one another). SFT will return misdiagnoses related to stochastic dependence when selective influence is not met (Eidels, Houpt, Altieri, Pei, & Townsend, 2011). However, if the stochastic dependence is the result of an external variable (Houpt, Blaha, McIntire, Havig, & Townsend, 2014, such as attention, see), SFT will return a correct diagnosis, as it still respects the assumption of selective influence.

Two possible operational states for two sub-processes creates a Double Factorial Paradigm, where four working conditions are possible: HH, where both sub-processes are treating optimal stimuli, LL, where both sub-processes are treating sub-optimal stimuli, and HL/LH, where one sub-process is treating an optimal stimulus and the other is treating a sub-optimal stimulus. In the analogy, a tired worker represents the L condition and a normal, rested worker represents the H condition. This experimental design allows us to measure the effect size of the change in processing capability, or the interaction contrasts of the four experimental conditions. Take for example a serial exhaustive architecture. The time taken by each sub-process is additive, assuming that both sub-processes are separate entities that have no overlap in individual RT (Sternberg, 1969). When a sub-process is in a Low condition, a certain amount of time is added to the overall processing time. When both sub-processes are in a Low condition then the additional time is counted twice. The overall completion time is therefore a reflection of which working condition each sub-process is in. There exists a variety of theories that use interaction contrasts including Mulligan and Shaw (1980), and more recently Cousineau, Donkin, and Dumesnil (2015).

SFT Methodology

As mentioned in the Nomenclature and Basic Principles section above, affecting the processing ability of each sub-process is mediated by the quality of the presented stimuli as well as selective influence. Figure 2 presents an example of the four conditions in an SFT analysis taken from an unpublished study of a same-different task designed specifically for an SFT analysis. A same-different task is one for which an individual must judge as rapidly and as accurately whether or not two successive stimuli are the same or different (Bamber, 1969).

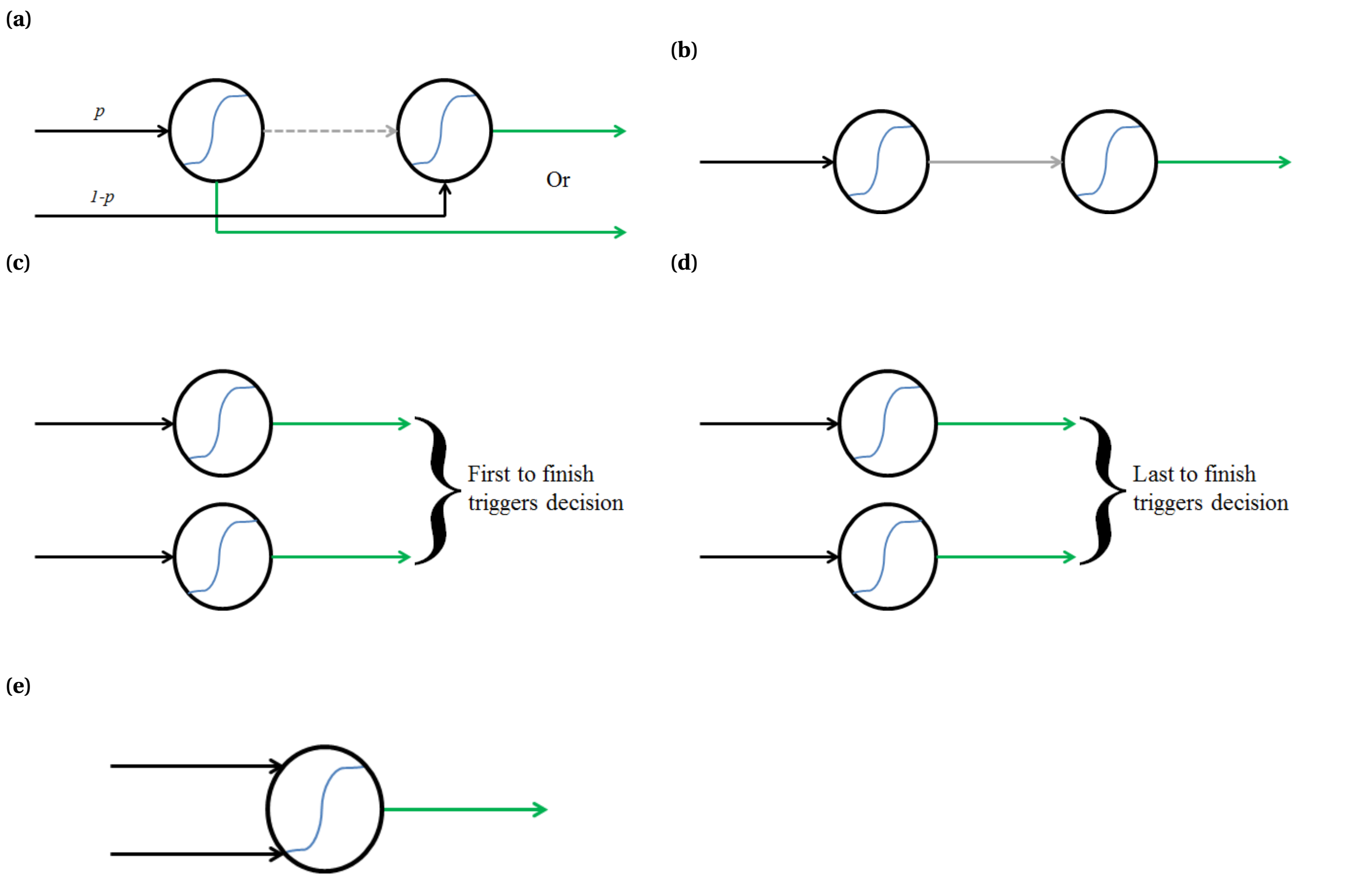
As can be seen, stimuli from the L condition are harder to discern than those from the H condition stimuli, and thus, it can be assumed that the processing for each will be different. Additionally, as the two stimuli are processed by different sub-processes, the fact that one letter is in the L condition has no effect on the other's processing capability (thereby respecting selective influence).

After gathering data for all four conditions, we can calculate the Mean Interaction Contrast (MIC), the first tool to determine the architecture between sub-processes. The MIC is measured using the following equation:

$$\text{MIC} = (\overline{RT}_{HH} + \overline{RT}_{LL}) - (\overline{RT}_{HL} + \overline{RT}_{LH}), \tag{1}$$

where $\overline{RT}_{HH}$ is the mean RT for the High-High condition, $\overline{RT}_{LL}$ is the mean RT for the Low-Low condition, $\overline{RT}_{HL}$ is the mean RT for the High-Low condition, and $\overline{RT}_{LH}$ is the mean RT for the Low-High condition.

Figure 1 ■ Schematics of the five different types of architectures that can be identified by SFT. a) serial self-terminating; b) serial exhaustive; c) parallel self-terminating; d) parallel exhaustive; e) coactive

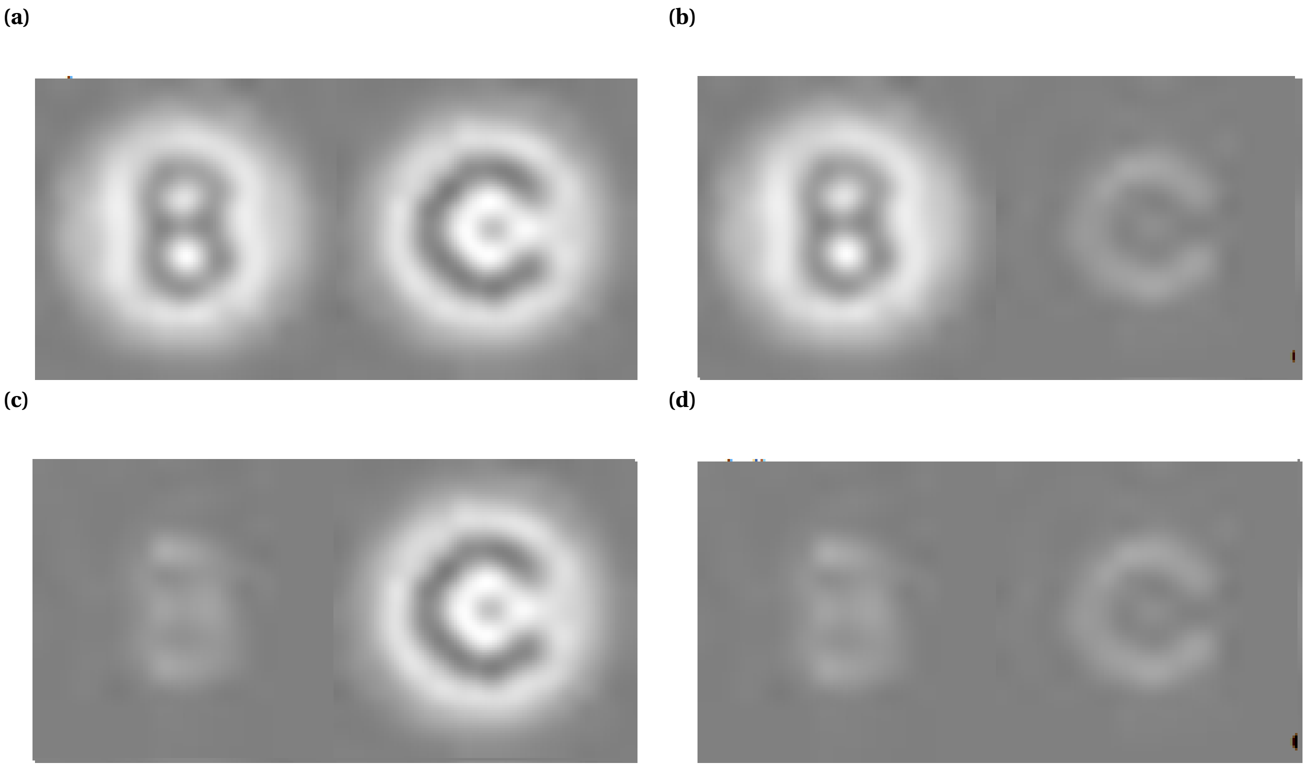

(a)
p
1-p
Or
(b)
(c)
First to finish triggers decision
(d)
Last to finish triggers decision
(e)

If the MIC is about 0, the architecture is considered serial. As stated earlier, this is related to the fact that the effect of each L condition is additive. If the MIC is positive, it is either parallel self-terminating or coactive and if the MIC is negative, then the architecture is parallel exhaustive. This relates to the fact that parallel architectures do not have additive L conditions. While the MIC provides some information regarding the interaction between sub-processes, the information provided is quite limited. For example, MIC cannot discriminate between the different stopping rules for serial architectures or detect differences between coactive and parallel self-terminating architectures. See Townsend and Nozawa (1995) for an in-depth proof of these claims.

Fortunately, to correct MIC's limitations, SFT expands on the study of contrasting the means to contrasting the entire distributions using the Survivor Interaction Contrast, or SIC. SIC relies on survivor functions of the RT data distributions. A survivor function (SF) is the opposite of the cumulative density function (CDF), another way of presenting the usual probability density function (PDF). PDFs are commonly used to represent a data distribution. For example the "bell-curve" line represents the PDF of the normal distribution. In contrast, the CDF represents the probability of sampling a score that is lower or equal than score t. For example, the median of a PDF distribution (score t) is represented as 0.50 in a CDF distribution, as there is a 50% chance of finding the median or a score below it. Figure 3 shows a PDF distribution (panel a), its associated CDF (panel b), survivor function (panel c), and cumulative hazard function, which will be discussed in the final section of the article (panel d; Luce, 1986).

As is seen, the survivor function simply shows the inverse of the CDF, or $S(t) = 1 - F(t)$, where $F(t)$ represents the cumulative function and $S(t)$ is the survivor function. Therefore, the SF returns the probability of finding a RT, or one greater than, at a certain time t. For example, the first quartile of a PDF distribution has cumulative probability of 0.25 as there is a 25% chance of finding a score below it. The SF of this quartile would then be 1 - 0.25 = 0.75; there is a 75% chance of finding a score that is higher than the first quartile.

Figure 2 ■ Examples of stimuli that create the four possible conditions needed for an SFT analysis inspired from an unpublished same-different task modified to be able to perform an SFT analysis on the results. a) High-High (HH) condition; b) High-Low (HL) condition; c) Low-High (LH) condition; d) Low-Low (LL) condition



Once we have the survivor functions for each of the four conditions, we can find the SIC using a formula based on the MIC expressed in (1):

$$\mathrm{SIC}(t) = (S_{HH}(t) + S_{LL}(t)) - (S_{HL}(t) + S_{LH}(t)) \qquad (2)$$

where SIC(t) is the result from the SIC at a given time t, $S_{HH}(t)$ is the score on the High-High survivor function at time t, $S_{LL}(t)$ is the score on the Low-Low survivor function at time t, $S_{HL}(t)$ is the score on the High-Low survivor function at time t, and $S_{LH}(t)$ is the score on the Low-High survivor function at time t. While SIC is similar to the MIC, SIC is a function that measures the interaction contrast of all t scores rather than a single summary value for the entire process.

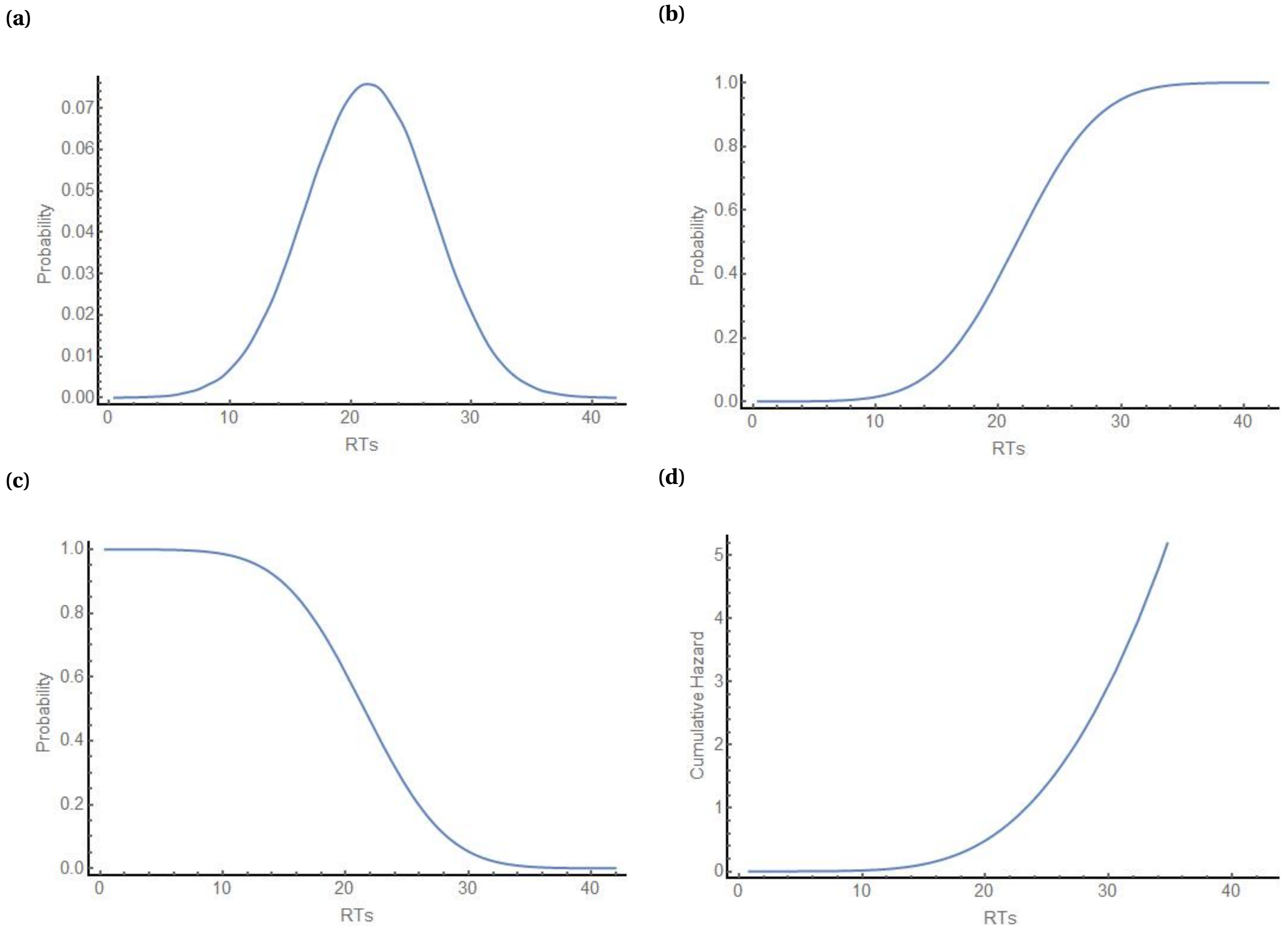
By plotting the series of results given by (2), we get a curve that can have five distinct patterns corresponding to the five processing architectures expressed above. The comparison of this curve to the theoretical curves given in Figure 3 pinpoints which processing architecture links two sub-processes.

Figure 5 shows the four survivor functions from a simulated dataset plotted together along with the associated SIC curve. The green lines linking both plots signifies the point for which (2) was performed on the survivor functions and the result of the calculation on the SIC curve. In this case, the SIC curve deems the architecture as being a serial exhaustive.

Even if the SIC curve offers a much clearer picture of which architecture is present, it is prudent to not rely on it alone. For example, consider a measured SIC curve that has a negative area followed immediately by a positive area. Following a comparison to the theoretical curves from Figure 4, one could conclude that the measured curve matches the theoretical one associated to serial exhaustive. One could also conclude that the measured curve somewhat resembles coactive's SIC curve. To quantitatively differentiate between the two architectures we can measure the MIC for both. If the MIC returns a non-null result, we can conclude that the architecture is coactive. If the MIC returns null, or near null, results we can conclude that the architecture is serial exhaustive.

It is also possible to visually differentiate between curves. Coactive's negative area must be smaller than the positive area. On the other hand, the positive and negative areas for the serial exhaustive curve are equal. Because MIC is in a fact a measure of the area under the SIC curve

Figure 3 ■ Three different ways of showing the same data in different types of probability distributions. a) PDF distribution, b) CDF distribution, d) SF distribution, d) CHF distribution



(where areas below zero are negative), this visual approach is the same as quantifying MIC.

Additionally, as introduced in Harding, Leblanc, Goulet, and Cousineau (submitted), SIC's centerline (measured as the median RT of all conditions pooled together) allows for a visual discrimination between architectures. The centerline of a plot always divides the two bumps at the point (or near the point) of intersection whereas it always cuts through the bumps for parallel architectures. Any centerline that does not follow these trends should lead the researcher to further investigate the architecture at play.

Table 2 synthesizes the possible results returned from an SFT analysis, which are adapted from Townsend and Nozawa (1995), Figure 2.

Using measures of capacity as an additional diagnostic tool

While we have explored MIC, SIC, as well as briefly introduce the SIC centerline, there exist other complementary diagnostic tools, the capacity indices which have also been used within a wide variety of applications (Yusuke, McCarley, & Kramer, 2015; Heathcote et al., 2015). These indices measure how and whether the addition of a second subprocess affects the overall processing capability of the system as a whole. Processing capacity can be interpreted as the amount of resources required by sub-process(es). Measuring capacity within the SFT framework therefore requires the addition of two more experimental conditions: where there is but a single sub-process that is working in an H or an L condition with the other one being completely neutral. For example, in the same-different task, it would be the equivalent of having two extra experimental conditions where a single letter is in the H condition with the other letter being completely absent (Houpt et al., 2014).

Capacity must be computed using distinct formulas for self-terminating and exhaustive architectures, using C_{OR} and C_{AND} measures respectively. In the standard C_{OR} measure presented in (3), one must use the cumulative hazard function (CHF) of RT scores. The CHF represents the cumulative of minus the Log of the survivor function.

$$\mathrm{C}_{OR}(t) = \frac{(H_{ab}(t))}{(H_a(t) + H_b(t))} \tag{3}$$

where $\mathrm{C}_{OR}(t)$ represents the capacity of the system, $H_a(t)$ represents the experimental condition at time t on the CHF when only sub-process a is in the H condition, $H_b(t)$ represents the experimental condition at time t on the CHF when only sub-process b is in the H condition, and $H_{ab}(t)$ represents the experimental condition at time t on the CHF when both sub-processes are in the H condition.

For the standard C_{AND} measure presented in (4), one must use the reverse cumulative hazard function (RCHF) of RT scores.

$$\mathrm{C}_{AND}(t) = \frac{(K_a(t) + K_b(t))}{(K_{ab}(t))} \tag{4}$$

where $\mathrm{C}_{AND}(t)$ represents the capacity of the system at time t, $K_a(t)$ represents the experimental condition at time t on the RCHF when only sub-process a is in the H condition, $K_b(t)$ represents the experimental condition at time t on the RCHF when only sub-process b is in the H condition, and $K_{ab}(t)$ represents the experimental condition at time t on the RCHF when both sub-processes are in the H condition.

In these measures of $\mathrm{C}_{OR}(t)$ and $\mathrm{C}_{AND}(t)$, a score of 1 indicates that the addition of a second sub-process has no effect whatsoever on the system as a whole (unlimited capacity). This means that the system has plenty of resources for all the sub-processes to work at their respective optimal rate. A score that is smaller than 1 indicates that the addition of a second sub-process hinders the overall processing capability of the system (limited capacity). This means that the system lacks resources and results in sub-processes not working optimally. Finally, a score that is above 1 indicates that the addition of a second sub-process improves the processing capability of the system as a whole (super-capacity). This signifies that both sub-processes help each other thereby requiring fewer resources to perform the task compared to a scenario where both sub-processes would work individually. For a more in-depth review of these capacity measures, see Townsend and Wenger, (2004), or Houpt and Townsend (2012).

Alternatively, Houpt and al. (2014) propose the following, non-standard, measures of capacity which makes similar predictions to those in (3) and (4). All values in the equations are equivalent to those presented in (3) and (4).

$$\mathrm{C}'_{OR}(t) = (H_{ab}(t)) - (H_a(t) + H_b(t)) \tag{5}$$

$$\mathrm{C}'_{AND}(t) = (K_{ab}(t)) - (K_a(t) + K_b(t)) \tag{6}$$

In these measures of $\mathrm{C}'_{OR}(t)$ and $\mathrm{C}'_{AND}(t)$, a score of 0 is indicative of unlimited capacity, a negative score indicates limited capacity, and a positive score indicates super-capacity. We encourage the use of these measures rather than the ones proposed in Townsend and Nozawa (1995) for two reasons. First, having an unlimited capacity reference point of 0 is more intuitive than a reference point of 1. Second, as the capacity is not a ratio, there are no divisions with zero.

Much like the SIC, the capacity measures are a function of time and can be plotted. Each capacity curve has a pattern that is characteristic of each processing architecture. These capacity curves are shown in Figure 6. In the left column we present the curves associated to the standard capacity measure (Equations 3 and 4) and in the right column we present the curves associated to the non-standard capacity measure (Equations 5 and 6).

Once plotted, the capacity curve of the architecture should match up with whatever architecture is proposed by the SIC. If, however, the SIC curve and capacity curves point to different architectures, it could be that an assumption of SFT (such as stochastic independence or selective influence) is not met. For a more formal review of capacity within an SFT analysis, see Houpt et al. (2014) where SFT computations are done with the R programming interface.

Conclusion

While Systems Factorial Technology is a useful tool to find the underlying architecture between two sub-processes in a cognitive paradigm, the methodology itself is not necessarily newcomer-friendly. In this article, we stripped down SFT into key points and presented a straightforward approach to the methodology. We presented the basics and nomenclature of a cognitive process and definitions of the possible architectures in a cognitive paradigm. Annexed is a Mathematica code that is ready to be implemented allowing for a visual aid on how the methodology diagnoses architectures. Table 3 presents and overview of the general methodology used with SFT.

Although SFT is well established in the literature, there is room for expansion. Most current SFT analyses focus on the 2×2 Double Factorial experimental design, yet some work has been done to further push the methodology to more complex experimental design with an arbitrary amount of sub-processes (Yang, Fific, and Townsend, 2013 where SIC curves are generalized to n sub-processes; and more recently Blaha and Houpt, 2015, where capacity measures are generalized to n sub-processes). While this article focuses solely on the simpler of the two, a review and tutorial of the more complex designs could be warranted.

The SFT methodology also has certain limits. For instance, model mimicking (and therefore SFT misdiagnoses) is possible when certain model components are altered. Eidels et al. (2011) showed that architectures could be misconstrued with one another when stochastic independence is altered. Harding, LeBlanc, Goulet, and Cousineau (submitted) explored accumulator models with variable thresholds and showed that SFT mistakenly diagnoses coactive architectures as being parallel self-terminating if the threshold's variability is sufficiently large. They further explore capacity coefficients as well as the SIC centerline to obtain a more complete diagnosis.

Finally, Tremblay, Harding, Chartier, and Cousineau (2014) recently applied the SFT methodology to the recall process of an architecture that is not one of the five detected by SFT, the Bidirectional Heteroassociative Memory, or BHM (Chartier & Boukadoum, 2006). The BHM uses a series of iterations of parallel units until activations stabilize to select response units. While not fitting any of the SFT's detected architectures, the stimuli are processed in an iterative fashion; the number of iterations is used as a measure of RT. Tremblay et al. (2014) found that the SIC curve resembles a serial exhaustive architecture, which is not at all in line with the parallel architecture actually used. This example allows us to observe the limits of SFT and warn us to not venture outside of the scope of the methodology as it may return results that are untrue, especially in the presence of atypical processes.

Although these limits add some uncertainty to the results of an SFT analysis, the methodology itself remains a staple, and an important one, in experimental psychology. SFT is an important step towards understanding the information processing of our brain and the more accessible SFT becomes the better.

Authors' note

We would like to thank Denis Cousineau, Vincent Leblanc, and Julien Lemay for their comments on the manuscript.

References

Bamber, D. (1969). Reaction times and error rates for "same" and "different" judgments of multidimensional stimuli. *Perception & Psychophysics*, *6*(3), 169–174. doi:10.3758/BF03210087

Blaha, L. & Houpt, J. W. (2015). An extension of workload capacity space for systems with more than two channels. *Journal of Mathematical Psychology*, *66*, 1–5. doi:10.1016/j.jmp.2015.01.002

Chartier, S. & Boukadoum, M. (2006). A bidirectional heteroassociative memory for binary and grey-level patterns. *IEEE Transactions on Neural Networks*, *17*(2), 385–396. doi:10.1109/TNN.2005.863420

Cousineau, D., Donkin, C., & Dumesnil, E. (2015). Unitization of features following extended training in a visual search task. In J. G. W. Raaijmakers, A. H. Criss, R. L. Goldstone, R. M. Nosofsky, & M. Steyvers (Eds.), *Cognitive modeling in perception and memory: a festschrift for richard m. shiffrin* (pp. 3–15). New York: Psychology Press.

Cousineau, D. & Shiffrin, R. M. (2004). Termination of a visual search with large display size effects. *Spatial Vision*, *17*(4), 237–352. doi:10.1163/1568568041920104

Donders, F. C. (1969). On the speed of mental processes. *Acta Psychologica*, *30*, 412–431. doi:10.1016/0001-6918(69)90065-1

Dzhafarov, E. & Schweickert, R. (1995). Decomposition of response times: an almost general theory. *Journal of Mathematical Psychology*, *39*(3), 285–314. doi:10.1006/jmps.1995.1029

Eidels, A., Houpt, J. W., Altieri, N., Pei, L., & Townsend, J. T. (2011). Nice guys finish fast and bad guys finish last: facilitatory vs. inhibitory interaction in parallel systems. *Journal of Mathematical Psychology*, *55*(2), 176–190. doi:10.1016/j.jmp.2010.11.003

Engmann, S. & Cousineau, D. (2013). Triple redundant signals effect in the visual modality. *Universitas Psychologica*, *12*(5), 1473–1488. doi:10.11144/Javeriana.UPSY12-5.trse

Fific, M., Nosofsky, R. M., & Townsend, J. T. (2008). Information-processing architectures in multidimensional classification: a validation test of the systems factorial technology. *Journal of Experimental Psychology: Human Perception and Performance*, *34*(2), 356–375. doi:10.1037/0096-1523.34.2.356

Fific, M., Townsed, J. T., & Eidels, A. (2008). Studying visual search using systems factorial methodology with target-distractor similarity as the factor. *Perception & Psychophysics*, *70*(4), 583–603. doi:10.3758/PP.70.4.583

Gaissmaier, W., Fific, M., & Rieskamp, J. (2011). Analyzing response times to understand decision processes. In A. K. Schulte-Mecklenbeck & R. Ranyard (Eds.), *M* (pp. 141–163). A Handbook of Process Tracing Methods for Decision Making. New York: Taylor & Francis.

Harding, B., LeBlanc, V., Goulet, M.-A., & Cousineau, D. (submitted). Applying systems factorial technology to discrete accumulators with varying thresholds. In D. R. Little, N. Altieri, M. Fific, & C.-T. Yang (Eds.), *Systems factorial technology: A theory driven methodology for the identification of perceptual and cognitive mechanisms* (pp. 1–43). San Diego, California: Elsevier Publishing.

Heathcote, A., Coleman., J. R., Eidels, A., Watson, J. M., Houpt, J. W., & Strayer, D. L. (2015). Working memory's workload capacity. *Memory and Cognition*, *43*(7), 973–989. doi:10.3758/s13421-015-0526-2

Houpt, J. W., Blaha, L. M., McIntire, J. P., Havig, P. R., & Townsend, J. T. (2014). Systems factorial technology with r. *Behavior Research Methods*, *46*(2), 307–330. doi:10.3758/s13428-013-0377-3

Houpt, J. W. & Townsend, J. T. (2010). The statistical properties of the survivor interaction contrast. *Journal of Mathematical Psychology*, *54*(5), 446–463. doi:10.1016/j.jmp.2010.06.006

Johnson, S. A., Blaha, L. M., Houpt, J. W., & Townsend, J. T. (2010). Systems factorial technology provides new insights on global-local information processing in autism spectrum disorders. *Journal of Mathematical Psychology*, *54*(1), 53–72. doi:10.1016/j.jmp.2009.06.006

Luce, D. (1986). *Response times: their role in inferring elementary mental organization*. New York: Oxford University Press.

Miller, J. (1982). Divided attention: evidence for coactivation with redundant signals. *Cognitive Psychology*, *14*(2), 247–279. doi:10.1016/0010-0285(82)90010-X

Mulligan, R. M. & Shaw, M. L. (1980). Multimodal signal detection: independent decision vs. integration. *Perception and Psychophysics*, *28*(5), 471–478. doi:10.3758/BF03204892

Sternberg, S. (1969). Memory-scanning: mental processes revealed by reaction-time experiments. *American Scientist*, *57*(4), 421–457.

Sternberg, S. (1998). Inferring mental operations from reaction-time data: how we compare objects. In D. N. Osherson, D. Scarborough, & S. Sternberg (Eds.), *An invitation to cognitive science, volume 4: methods, models, and conceptual issues* (pp. 365–454). Cambridge, MA: MIT Press.

Townsend, J. T. & Ashby, F. G. (1983). *The stochastic modeling of elementary psychological processes*. Cambridge, MA: Cambridge University Press.

Townsend, J. T., Fific, M., & Neufeld, R. W. J. (2007). Assessment of mental architecture in clinical/cognitive research. In T. A. Treat, R. R. Bootzin, & T. B. Baker (Eds.), *Psychological clinical science: papers in honor of richard m. mcfall* (pp. 223–258). New York: Psychology Press.

Townsend, J. T. & Nozawa, G. (1995). Spatio-temporal properties of elementation perception an investigation of parallel, serial, and coactive theories. *Journal of Mathematical Psychology*, *39*(4), 321–359. doi:10.1006/jmps.1995.1033

Treisman, A. M. & Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology*, *12*(1), 97–136. doi:10.1016/0010-0285(80)90005-5

Tremblay, C., Harding, B., Chartier, S., & Cousineau, D. (2014). System factorial technology applied to artificial neural network information processing. In O. B. & S. a. L. (Eds.), *Goertzel* (pp. 258–261). Springer International Publishing: Artificial General Intelligence. doi:10.1007/978-3-319-09274-4_29

Yang, H., Fific, M., & Townsend, J. T. (2013). Survivor interaction contrast wiggle predictions of parallel and serial models for an arbitrary number of processes. *Journal of Mathematical Psychology*, *58*, 21–32. doi:10.1016/j.jmp.2013.12.001

Yusuke, Y., McCarley, J. S., & Kramer, A. F. (2015). Workload capacity across visual field in young and older adults. *Archives of Scientific Psychology*, *3*(1), 62–73. doi:10.1037/arc0000016

Appendix A: Appendix A: Visualize an SFT analysis using Mathematica

This section serves as an applied introduction and example of the System Factorial Technology methodology. Attached to this article is a Mathematica notebook that has all of the code presented here. In this notebook each command is outlined and described to allow new Mathematica users to get up to speed quickly.

Here, the four SFT experimental conditions' RT will be represented by `RThh, RThl, RTlh, RTll` (h represents the H condition and l represents the L condition). Prior to any SFT analysis, erroneous responses and outliers should be removed. In addition, analyses should be performed on a within-subject basis to see individual architectures participants employ when doing the experiment.

The data we are working with

In this first section we plot all four conditions' distributions in a single plot. We believe that this is an essential step to any SFT analysis and should be done prior to anything else. Here we present the PDF (where distributions are drawn with a line) of each distribution on a single plot, the bin sizes are set with `Automatic`.

```
SmoothHistogram[{RThh,RThl,RTlh,RTll},Automatic,"PDF",
AxesLabel->{"RT","Frequency"}]
```

If one wants to observe the distributions individually, it is possible using this alternative command, where a histogram will be used instead of a line:

```
Histogram[RThh,Automatic,"PDF"]
```

The Mean Interaction Contrast (MIC) and its statistical tests The MIC is obtained in Mathematica, by measuring the mean for each condition using (1).

```
MIC=(Mean[RThh]+Mean[RTll])-(Mean[RThl]+Mean[RTlh])
```

For the purpose of statistical testing, MIC can be tested using the same tools as a single sample of RT; the standard error however, is given by the four conditions' standard errors with the following relation:

$$SE_{MIC} = \frac{1}{2}\sqrt{SE^2_{RTll} + SE^2_{RTlh} + SE^2_{RThl} + SE^2_{RThh}} \tag{7}$$

The test of the null hypothesis that MIC = 0 is therefore obtained with:

```
SE[data_]:=StandardDeviation[data]/Sqrt[Length[data]]
SEmic=1/2 Sqrt[SE[RTll]^2+SE[RTlh]^2+SE[RThl]^2+SE[RThh]^2];
pmic=2(1-CDF[NormalDistribution[],Abs[MIC/SEmic]])
```

where `pmic` returns a p value based on a z test.

The Survivor Interaction Contrast (SIC) Curves and the Kolmogorov-Smirnov test

To calculate the SIC, one must first transform the data into survivor functions. This is done by reprising the distribution command above and replacing `"PDF"` with `"SF"`. Again, it is important to visualize the curves to gather intuitions on how the model is built. The survivor function of each experimental condition is named sXX, where X represents which condition each sub-process is in (ex: the SF of RThh is shh). This must be repeated for each experimental condition.

```
shh=HistogramList[RThh,bins,"SF"][[2]];
...
ListPlot[{shh,shl,slh,sll},Joined->True,AxesLabel->{"RTs","Frequency"}]
```

The HistogramList function works identically as the Histogram function, except that it produces two lists instead of a single plot. The first list is a vector of the bins' limits; the second list is the survivor function's value at those points. It is necessary to use the same custom-defined bin sizes for all four conditions to make sure that four lines are properly aligned. For this reason, custom bin sizes were specified manually rather than using the Automatic option; the simplest specification is to use `bins={lo, hi, step}` where bins are of size "step" (often 1), and are evenly spaced between "lo" and "hi". We found the best results were to specify lo as zero and step as 1 and use the maximal RT value across all four conditions to compute the hi value:

```
hi=Max[Flatten[{{RThh,RThl,RTlh,RTll}}]];
bins={0,hi,1};
```

Once the data are available and plotted, we can compute SIC using:

```
SIC=(shh+sll)-(shl+slh);
```

This operation will automatically be performed for all times contained within the sXX lists. Once measured, we plot the SIC using the following command:

```
ListPlot[SIC,Joined->True,PlotRange->{All,{-1,1}},
AxesLabel->{"t ","SIC(t)"},GridLines->{{{centerline,{Red,Dashed}}}}]
```

As seen, we inserted a single vertical gridline at the "centerline". This refers to the SIC centerline measure proposed by Harding et al. (submitted) that can be used as an additional diagnosis tool. The centerline is the median of all four experimental conditions pooled together if all experimental conditions have the same amount of trials. The centerline is computed with:

```
centerline=Median[Flatten[{{RThh,RThl,RTlh,RTll}}]];
```

Statistical test of a SIC curve can be accomplished using the Kolmogov-Smirnov test adapted to the present contrast (Houpt & Townsend, 2010). To do so, we first measure the harmonic mean of the sample sizes:

```
hMean=HarmonicMean[{Length[RThh],Length[RTlh],
Length[RThl],Length[RTll]}];
```

The Kolmogov-Smirnov test tests the null hypothesis that the SIC is never significantly different from zero; it returns a D test statistic and p value. However, as SIC curves can be positive and negative on the same curve (as with the serial exhaustive and coactive architectures), we must carry two different tests, one for positive portions of the SIC and one for negative portions of the SIC. The positive portion of the SIC curve is tested with:

```
Dplus=Max[SIC]
Pplus=Exp[-2 hMean/4 (Dplus)^2]
```

whereas the following tests the null hypothesis that the SIC is never significantly below zero:

```
Dneg=Min[SIC]
Pneg=Exp[-2 hMean/4 (Dneg)^2]
```

Measuring and plotting the capacity indices

As mentioned in the text, capacity indices give additional diagnosis power to SFT. Capacity is measured using one of two measures depending on which architecture is found using the MIC and SIC.

To plot the standard measures proposed in Townsend and Nozawa (1995), one must first have access to the RT of both sub-processes working alone (RTh1 and RTh2) versus when they are working together (RThh). In the code presented here we assume that users have gathered data from sub-processes working in the H condition.

When the diagnosis points to self-terminating architectures, the C_{OR} measure must be used. As aforementioned, this measure uses the cumulative hazard function of the RT scores which are calculated using the following code (CHF simply replaces SF from the survivor functions code).

```
Hab=HistogramList[RThh,bins,"CHF"][[2]];
```

...

Here we only show the RThh condition but this measure needs to be repeated with RTh1 and RTh2. $\mathrm{C}_{OR}(t)$ is then measured and plotted using Townsend and Nozawa (1995) measure of capacity:

```
CapOR=Hab/(Ha+Hb);
ListPlot[CapOR,Joined->True,PlotRange->{All,All},
AxesLabel->{"t ","C_(OR)"}]
```

Alternatively, we can also measure and plot Houpt et al. (2014, 's) non-standard measure of capacity using:

```
CapOR1=Hab-(Ha+Hb);
ListPlot[CapOR1,Joined->True,PlotRange->{All,All},
AxesLabel->{"t ","C_(OR)"}]
```

When the diagnosis points to exhaustive architectures, the C_{AND} measure must be used. As this measure uses the reverse cumulative hazard function and Mathematica does not have a specification for it, we must calculate it manually using the following code.

```
Kab=HistogramList[RThh,bins,"CDF"][[2]];
Ka=HistogramList[RTh1,bins,"CDF"][[2]];
Kb=HistogramList[RTh2,bins,"CDF"][[2]];
{Ka,Kb,Kab}={Ka,Kb,Kab}//Log;
```

The last line replaces all values of Ka, Kb, and Kab on the CDF with their Log values. $\mathrm{C}_{AND}(t)$ is then measured and plotted using:

```
CapAND=(Ka+Kb)/Kab;
ListPlot[CapAND,Joined->True,PlotRange->{All,All},
AxesLabel->{"t ","C_(AND)"}]
```

Alternatively, we can also measure and plot Houpt et al. (2014)'s non-standard measure of capacity using:

```
CapAND1=Kab-(Ka+Kb);
ListPlot[CapAND1,Joined->True,PlotRange->{All,All},
AxesLabel->{"t ","C_(AND)"}]
```

Appendix B: Generating simulated RTs based on a given architecture

Here we define the distribution for each of the two sub-processes in each stimulus intensity condition. The commands create simulated RT based on the normal distribution for convenience only. For more realistic simulations, other distributions should be used (for example, Houpt et al., 2014, , used Weibull distributions with shape parameter of 2). To simulate real-world data, a sub-process in a Low intensity condition must have slower RT than in the High intensity condition. Here, for parsimony's sake, the standard deviation for each distribution is the same for all conditions but can be easily changed to be different between all conditions if one would want.

```
D[m_,s_]:=NormalDistribution[m,s];
mh1=20;
ml1=36;
mh2=23;
ml2=40;
s=5;
n=50000;
```

In all the following simulations, we generated 50,000 simulated RTs to ensure clean curves and a reduced level of noise. If one would want to replicate more realistic SFT results, we encourage lowering the n to more realistic sample sizes that are in the hundreds rather than in the tens of thousands.

Model Simulations

The approach taken here is to first generate all of the sub-processes' times separately, hence yielding a list of n processing times for the two sub-processes involved. These lists are then combined using a specific aggregation function depending on the processing architectures, which returns the time taken in a specific experimental condition. For example, in a HH condition the individual times of each sub-process are simulated with:

```
time1=RandomVariate[D[mh1,s],n];
time2=RandomVariate[D[mh2,s],n];
```

The processing times of RThh can then be found with:

```
RThh=MapThread[Plus,{time1,time2}];
```

in which `Plus` is the aggregation function used to join all the entries of time1 and time2. This aggregation represents a serial exhaustive architecture as the time for both sub-processes to finish is added together for the total completion time. With parallel models, simply replace `Plus` with `Max` and `Min` for exhaustive and self-terminating models respectively.

Regarding serial self-terminating architecture, it is not known which sub-process performs the task, unless we have access to information regarding which sub-process performs first. We must therefore assume that it is indeed random between trials. The following aggregation functions choose randomly between both sub-processes, based on a probability p:

```
ChooseOne[time1_,time2_,p_]:=If[Random[]<p,time1,time2];
```

Because this function has three parameters, they must be explicitly given in the MapThread function of each of the four experimental conditions. Individual times of each sub-process are simulated with:

```
time1=RandomVariate[D[mh1,s],n];
```

```
time2=RandomVariate[D[mh2,s],n];

RThh=MapThread[ChooseOne[#1,#2,1/2]&,{time1,time2}];
```

in which # 1 and # 2 are placeholders for the entries of time1 and time2 in that order.

For coactive processes, we must assume that sub-processes are not sending a single "completed" message, but sequentially many mini-activations, or sub-tasks, that are collected (pooled) until k of them have been received:

```
Coactive[time1_,time2_,k_]:=Sort[Join[Accumulate[time1],
Accumulate[time2]]][[k]]
```

The threshold parameter is set with the following code. Indicates how many sub-tasks are needed, irrespective of which process completed them. It is set with:

```
k=10;
```

Much like the rest of the architectures, coactive's aggregation function takes two lists of k number of RT (here labeled time1 and time2), joins them together, and the kth fastest one is then sampled (the remaining k times are unneeded).

```
time1=RandomVariate[D[mh1],s],{n,k}];
time2=RandomVariate[D[mh2],s],{n,k}];
RThh=MapThread[Coactive[#1,#2,k]&,{time1,time2}];
```

For capacity measures, one must slightly vary the code to simulate a single sub-process working at a time. This is done by sampling k times each sub-processes' distributions and taking the last score that was sampled.

```
RTh1=Table[Accumulate[RandomVariate[D[mh1,s],{k}]]//Max,{n}];
RTh2=Table[Accumulate[RandomVariate[D[mh2,s],{k}]]//Max,{n}];
```

Citation

Harding, B., Goulet, M. A., Jolin, S., Tremblay, C. Villeneuve, S. -P.,& Durand, G. (2016) Systems Factorial Technology Explained to Humans. *The Quantitative Methods for Psychology*, *12*(*1*), 39-59.

Received: 22/05/2014 ~ Accepted: 26/11/2015

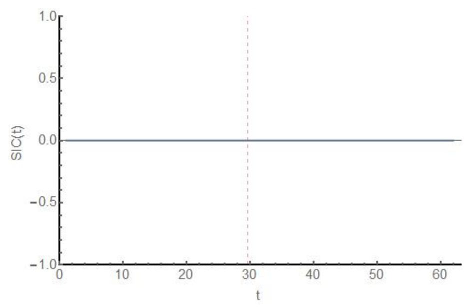
Tables and Figures follows on next page

Table 1 ■ SFT Cheat Sheet

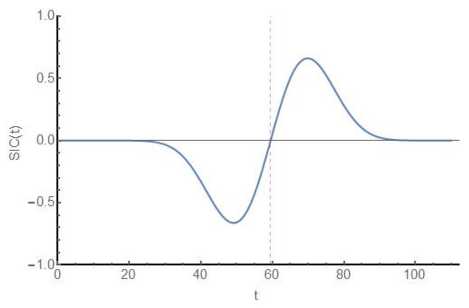
Cognitive Process	Specialized aspect of the brain treating specific parts of our deep and complex cognitive world, to make sense of it all.
Sub-process	Focalized part of a cognitive process specialized for specific tasks. Ex: detecting shapes, sizes, movements.
Processing order	Way for which both sub-processes are organized. They can fire sequentially (serial) or simultaneously (parallel) to trigger a decision.
Stopping-rule	The way for which a sub-process fires and decides when to trigger a decision. Can be when both sub-processes have finished (exhaustive) or when a single sub-process finishes (self-terminating).
Architecture	Combination of processing order and stopping rule. There are five standard architectures as detected by SFT.
Serial self-terminating	Sequentially placed sub-processes triggering a decision when one has finished. Decision is triggered by the first sub-process in line. The order of the sequence is inherently random.
Serial Exhaustive	Sequentially placed sub-processes only triggering a decision once both sub-processes have finished. The total time for this architecture is the individual time of each sub-process added together. The individual times of each sub-process are inaccessible.
Parallel self-terminating	Sub-processes accumulating information simultaneously where a decision is triggered as soon as one of the sub-processes finishes. Decision is based on the fastest of both sub-processes. The identity of the sub-process that triggered the decision is inaccessible.
Parallel exhaustive	Sub-processes accumulating information simultaneously where a decision is triggered as soon as both of the sub-processes finish. Decision is based on the slower of both sub-processes. The identity of the sub-process that triggered the decision is inaccessible as is the time of the sub-process that finished first.
Coactive	Similar to the parallel self-terminating, the decision is made once a enough (k) mini-activations are present. However, both sub-processes work towards a single goal. The individual sub-processes' times and contribution towards the decision making process is inaccessible.
Selective influence	Key to SFT where one must select sub-processes and affect their processing rate without affecting the rest of the cognitive process as a whole. The rest of the processes are then considered as constants and cancelled out.
Double Factorial Paradigm	Experimental design where two operational states (H and L) are possible for two sub-processes (A and B) which have been targetted using selective influence.
Mean interaction contrast (MIC)	Simple equation measuring the effect size between means of the four conditions as required by SFT. It is measured with: MIC = $(\overline{RT}_{HH} + \overline{RT}_{LL}) - (\overline{RT}_{HL} + \overline{RT}_{LH})$.
Survival interaction contrast (SIC)	Equation similar to the MIC, where the interaction contrast is measured for each moment in time (along the X-axis of the plot) between all four conditions. Plotting the result of the function gives the SIC curve. It is measured with: $\text{SIC}(t) = (S_{HH}(t) + S_{LL}(t)) - (S_{HL}(t) + S_{LH}(t))$.
SIC Curve	Telltale curve for which each specific architecture has a distinct curve associated to it. Matching of the measured SIC curve and theoretical SIC curve is how the detection of an underlying architecture linking two sub-processes is made.
Capacity indices	Measure of a system's processing capability when sub-processes are working together versus when they are working seperately. Depending on the stopping-rule found in the SIC, two different measures of capacity are used. When the architecture is self-terminating use $\text{C}_{OR}(t) = \frac{(H_{ab}(t))}{(H_a(t)+H_b(t))}$ or $\text{C}'_{OR}(t) = (H_{ab}(t)) - (H_a(t) + H_b(t))$. When the architecture is exhaustive, use $\text{C}_{AND}(t) = \frac{(K_a(t)+K_b(t))}{(K_{ab}(t))}$ or $\text{C}'_{AND}(t) = (K_{ab}(t)) - (K_a(t) + K_b(t))$

Figure 4 ■ The five detected architectures by SFT. The red dotted line represents the SIC centerline. a) SIC curve for a serial self-terminating architecture; b) SIC curve for a serial exhaustive architecture; c) SIC curve for a parallel self-terminating architecture; d) SIC curve for a parallel exhaustive architecture; e) SIC curve for a coactive architecture.

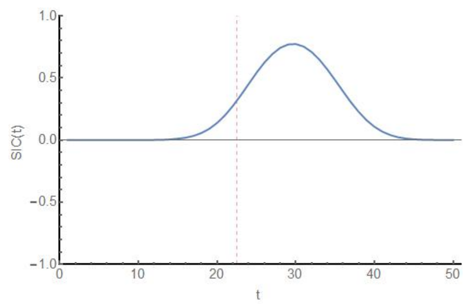
(a)



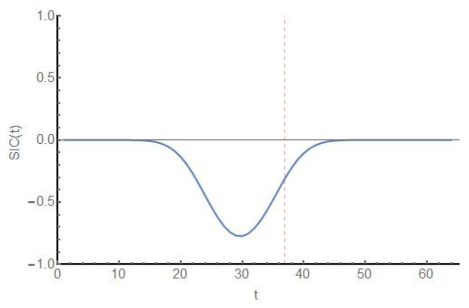
(b)



(c)



(d)



(e)

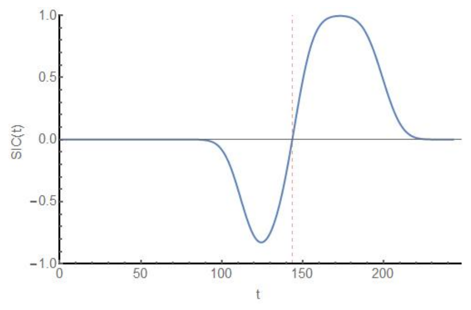


Table 2 ■ Summary of results returned in a typical SFT analysis. Details have been adapted from Table 2 in Townsend and Nozawa (1995).

Architecture	MIC	SIC curve trend
Serial Self-Terminating	Null	Flat Line on Null
Serial Exhaustive	Null	Negative Bump → Positive Bump (both areas are equal)
Parallel Self-Terminating	Positive	Positive Bump
Parallel Exhaustive	Negative	Negative Bump
Coactive	Positive	Negative Bump → Positive Bump (Negative Bump is smaller)

Figure 5 ■ Four survivor curves of each condition plotted together along with the associated SIC curve. The green lines linking both plots are the points for which Equation (2) was performed on the survivor functions and the result of the calculation on the SIC curve. In this case the curve represents a serial exhaustive architecture.

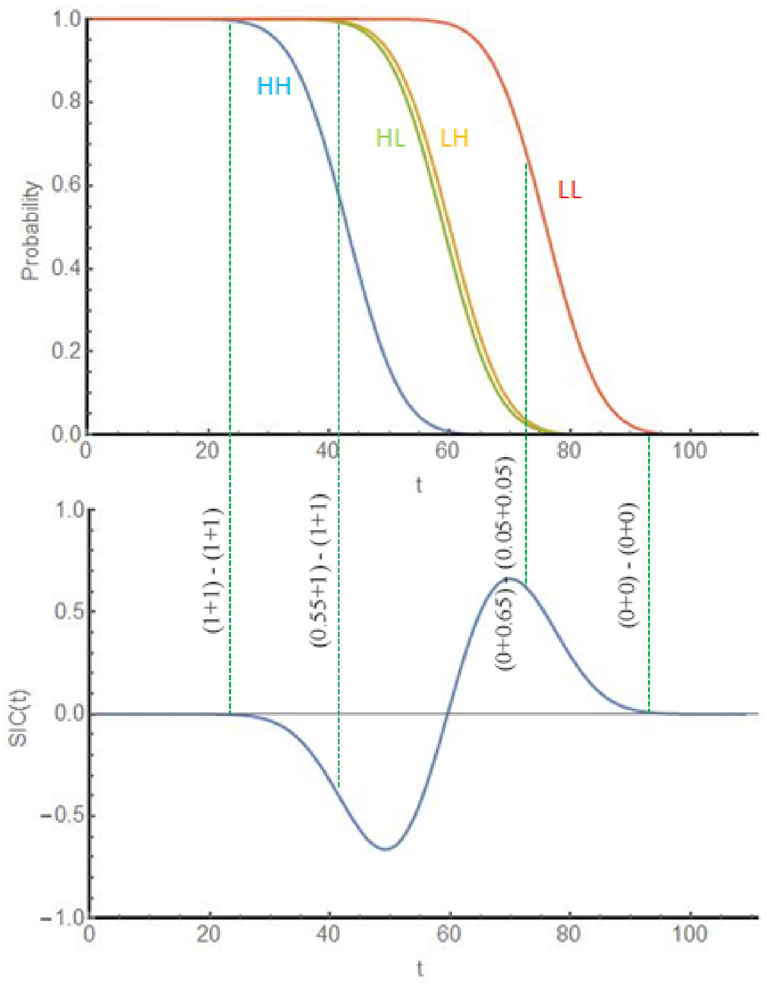
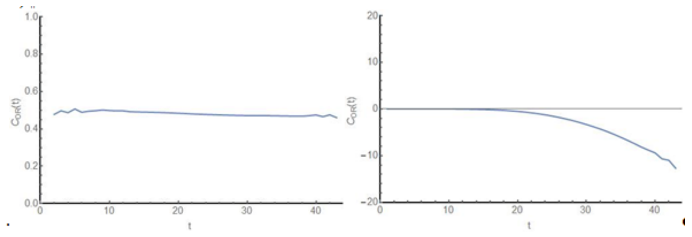
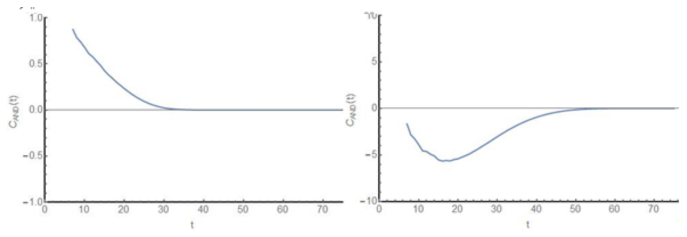


Figure 6 ■ Capacity curves for each of the five architectures. C_{OR} and C_{AND} measures were used depending on the architecture's stopping-rule. The left column represents the standard capacity measure introduced in Townsend and Nozawa (1995) and the right column represents the alternative, non-standard capacity measure presented in Houpt, Blaha, McIntire, Havig, and Townsend (2014). a) serial self-terminating, b) serial exhaustive, c) parallel self-terminating, d) parallel exhaustive, and e) coactive.

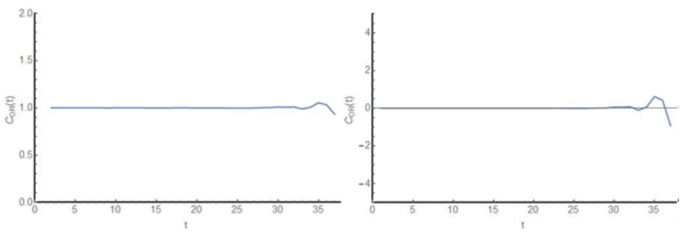
(a)



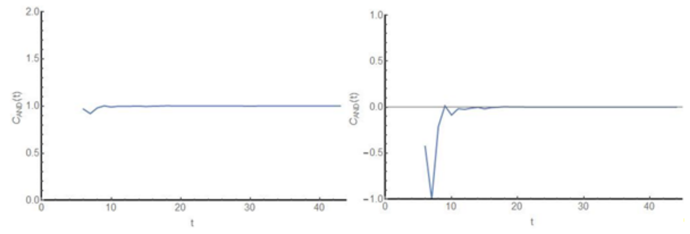
(b)



(c)



(d)



(e)

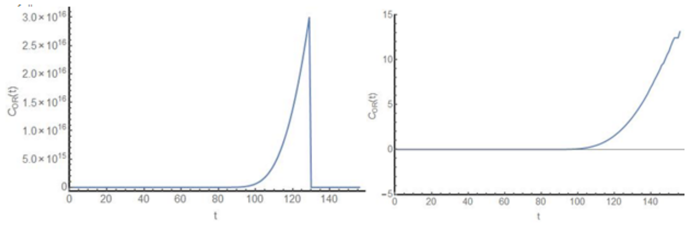


Table 3 ■ Abbreviated Methodology of SFT

	To obtain SIC and MIC
1.	Meet the selective influence assumption by picking a manipulation of the conditions that will only affect a single sub-process at a time;
2.	Affect the stimuli to create High and Low conditions for each of the two sub-processes;
3.	Create the four experimental conditions using a 2×2 design;
4.	Test and gather RT for each of these four conditions;
5.	Gather the mean for each condition and measure the MIC;
6.	Transform the PDF distributions into SF distributions;
7.	Measure and plot the SIC(t);
8.	Compare the measured SIC curves to the theoretical SIC curves to interpret the underlying architecture between the two sub-processes (consult Table 1);
	To obtain capacity curves
1.	Pick manipulations so that one sub-process is not operating;
2.	Affect the stimuli to create Hx, xH, and HH conditions where x denotes a non-operational sub-process;
3.	Combine stimuli in a 3 condition design;
4.	Test and gather RT for each of these three conditions;
5.	Transform the PDF into CHF or reverse-CHF depending on the stopping-rule found in the SIC;
6.	Measure capacity from all 3 functions;
7.	Plot the resulting capacity curve;
8.	Interpret the architecture by comparing the measured capacity curves to the theoretical capactiy curves (consult Table 1);