From AI theory to APMs and beyond:

7 things to watch for on Systems of Intelligence

There is no problem. *Yet*

In 2007, Dick Cheney used a pacemaker with wireless communication capabilities, and forever shamed the lack of imagination of early AI critiques.

Since then, AI evolved through leaps and bounds, confined only symbolically by a combination of physics, ethics and laws too expensive, controversial, or slow to keep up with the imagined solutions to not-yet-existing problems. Like a three-headed human (it can happen, too, it's called *polycephaly*), the theory, system design, and application development grew together but in parallel and—although sporadically shortcircuiting along the way—kept moving forward as pushed by an intrinsic need to keep searching.

Operating according to proprium rules, schedules, and patterns, the three areas somehow managed to make enough an individual dent in the society-at-large that together they pushed us all into what now is clearly the golden age of artificial intelligence. The one thing they all seem to have in common (besides being designed and run by really, really smart people) is this: once every decade or so, each is labored by the birth of a completely new class of concepts. They're driven not by an existent need, but by the luring capacity of predictive modeling to solve theoretical problems that in turn may help first discover solve some other problems, closer to home in object and timing but entirely out of view. (In other words: let's find the hardest thing to prove, use machines to demonstrate it, then use the results to identify real problems, and then go back to machines to solve those. Because, you know, we now can.)

What happens between the (overall) theory and the (individual) practical applications is where intelligence seems to lie. Not in the power of searching and analyzing huge spaces (which machines can and humans can't), but in establishing the direction of the search, and using the results in a manner that's useful, easy to manage, and performant.

And while theory can insofar be only generated by humans and applications can theoretically be 100% machine-ran, designing systems capable of machine learning, adaptable performance management, and some degree of self-assistance may well be the next step in meeting Mr. Data.

Systems of Intelligence: a *Systema Naturae*

Definitions

Down to its bare bones, the concept of system is easily identifiable and readily transferable from nature into the realm of the man-made; for both, a system is a collection of elements that function together and ring true all of the following:

1. The sum behavior of the system displays features that can't be realized by summing the behaviors of the isolated components.
2. The relationships between the parts are more important than the characteristics of the individual components.
3. They are dynamic; changes can reflect in functions as much as in goals.
4. The system boundaries are flexible and evolve with the scope of the system.

In technology, the syncretic character of human constructs lacks situational intelligence or pragmatic intelligence- but makes up more than enough in the area of contextual intelligence, no matter how dynamic the settings.

Contextual intelligence is built upon a system's ability to:

1. Reason- although not the first thing that comes to mind (no pun intended), reasoning is a set of processes that allows a system to create a basis for analysis, directed decisioning, and predictions (for example: if all is true for a class of things in general, it is true for all the members of that class).
2. Learn- either through direct collection of data or by being taught, systems can acquire knowledge by way of auditory learning, linear episodes, perceptual learning, relational comparisons, or geolocation.
3. Perceive information- the process of acquiring, selecting, cleaning, organizing and interpreting sensory data. In AI, perception also includes following the rules of meaningfulness.
4. Solve problems- often conditioned heavily by limitations to the above, problem involves using the available resources to move from the current state to a desired state within given parameters.

Moving from a contextually intelligent system (which we've already achieved through technologies such as APMs, for example) to a scope-specific system of intelligence and then to an universal system of intelligence may be dependent on special learning abilities (like free observation and linguistic learning), creativity, or decision-making.

Some of these are features so inherently human that replicating them would probably push us from *AI* straight into *AH* (*artificial human*); in the meantime, scope-specific systems of intelligence will make a perfectly fine line to cross.

The evolution of intelligence

The prehistoric species/ BI

In the past, the shiniest, brainiest intelligence came via business intelligence systems operating on data reservoirs much like someone looking for a book in a library. Need for knowledge would be translated into the right question to ask (info on the flu would be found in a medical material, for example), pointed in the right direction (a librarian), expressed (asked out loud) and, if all goes well, the librarian would take their time looking for the right book and bring it back while behind you a line starts forming.

As data became Big Data and BI became more important in making business decisions, the pace at which the decisions needed to be made also increased- posing an exponential burden on legacy systems designed to search, and not to think. Storage capacity can be increased but it's expensive, and the more data is stored the slower the results. The slower the result, the less timely the insight and therefore more irrelevant the business decisions it's trying to inform. And assuming none of this is an issue, accuracy might be. Traditional BI-based intelligence systems create insights based only on stored data, while potentially relevant information is, well, made to wait in line for its turn with the librarian. Even with advanced analytics, exploratory perspectives on data are not deep enough or fast enough to, say, inform real-time customer experience or prevent a cyber attack.

Legacy systems, when highly-specialized and flawlessly integrated with other technologies on the data-from-source-to-action continuum, can enrich decision making with valuable historical insight and, with a bit of luck, provide performance management metrics that are accurate for a little while; but that's rare.

Most businesses today need immediate, contextually-relevant actions but can't get them because their systems suffer from inherited parasites: antiquated BI solutions, cumbersome development tools, dead-end code.

The contemporaries/ OIs and APMs

Current methodology is following one of two paths: either previse systems with solutions to imagined problems in an attempt to preempt them (and gain the bragging rights of being the first), or try to perfect existing systems by making them bullet-proof to the known issues at hand. Both approaches are prone to eventual inadequacy, because of 3 recurrent themes in today's designs:

- **Uniqueness:** from proprietary languages to custom solutions built on open-source, systems are limited by arguably unimaginative need-at-hand scenarios, with very little attention given to integration issues or universality, for that matter. Add to that budgets limiting design to addressing only the immediate, and the result is usually a solution that is uber-performant but highly specialized—therefore smart, but hardly intelligent.

- **Real-time-ness:** as iterated many times by the likes of Gartner and Forrester, the value in data (and the insight it contains) is perishable- and the more time passes, the less actionable insight is—particularly for industries or application categories that depend on real-time action: online retail, new-wave telecommunications, cybersecurity, industrial internet.

  The sheer processing power of new technologies got us to a point where we can either predict or detect anomalies with millisecond precision, and teach our systems to react according to preset formulas so that we minimize the negative impact: network sections are shut off and bypassed, identity theft attempts detected and alerts sent, and so on.

- **Hybridization**: like a red blood cell, from the moment a data point is produced to the moment it goes to rest (having spewed out whatever intelligence it contains, separately or in context), it moves through a puzzle of technologies stuck together like organs in a body. But as you have it, sometimes life is better with an artificial replacement, and a part or integer engineered for a particular function will almost always perform better than a non-adaptive solution, no matter how natural it seems. Something like a stream processing platform integrates Fast and Big Data better than, say, a Lambda model; a cochlear implant produces purer sounds than a good old traditional ear drum.

> Moreover, within the same technology there's a high chance various functionalities have been added atop an initial base, and patched together through a crispr-like exercise meant to streamline. And, by the rule of technology law, the more connections, the more points of vulnerability.

With so many drawbacks to the default design, hopes that a system doesn't have any bugs, the hardware never fails, or the loads will never ever be too much are naive. Stream processing can analyze millions of data points per second, but can't control the fact that devices are easy to be hacked before their first peep- compromising the integrity of results before the birth of the first record. Similarly, applications can be developed perfectly, but entirely depend on the processing system feeding them and—if they're consumer-destined—on the mere capacity of the device they're hosted on or—why not?— event the end user. (Imagine, for example, a smart system that can interpret particular usage patterns of someone who, say, is color-blind, and adjust its visualization tools to increase contrast or proportions for a better experience).

The dawn of SoIs

Self-defending, *self-healing, self-perfecting*

Long is the road to a life-like system capable of both inductive and deductive reasoning, complete with observational learning skills and able to perceive information by way of patterns—but we're slowly getting closer by finally embracing the inevitable: there is no such thing as perfection. **Embracing the inevitable simply means designing systems able to predict likely future (analytics- check), deal with issues (automated actions- check), recuperate (in progress) and learn from each event enough to either avoid similar failures completely, or self-prepare so that the effects will be negligible (no clue).**

Today's intelligent systems are already operationally smart, and currently focus on perfecting business logic within the applications themselves (and with over 63% of the total being consumer-apps, that's rather wise). Airlines, banks, telecom companies correlate different events as they happen, and use the real-time insight to trigger immediate actions: fraudulent attempts are stopped and signaled mid-transaction, insurance rates are adjusted automatically based on driving behavior, and so on. But does that mean the system is learning?

The problem is this: even with advanced predictive capabilities, the process is doomed to be both **reactive** and **repetitive**: a deviation from what was established to be the desired window of normal is met with a certain, pre-programmed action, likely the same if the input doesn't vary and subsequent conditions are not set. Even the most advanced robot needs thousands of throws until it manages to catch a ball; it can visually estimate and measure trajectories with sub-second latency, and still not deduce that, if the thrower changes hands, it needs to adjust position ever so slightly for an easier catch. And most certainly we can't expect it to read intention to change hands based on previously-displayed body movement patterns.

In other words, we've still got work to do to move from a data-driven/reactive/assisted model to a context-driven/proactive/self-sufficient one—but with the advent of AI, machine learning, and using both for context analytics+performance monitoring, we're well on our way.
Measured by time-to-change and how apparent the triggers, here are the three levels of responsiveness in the pyramid of intelligent, responsive, elastic systems

Self-defending: detecting + blocking
There are three levels at which a system can exercise security:

- Data: veracity issues, incompleteness, or physical hacking of devices before data even enters a system can be immediately detected and flawed data can be either repaired, reconstructed, or eliminated before it enters the analytical phase. Furthermore, an intelligent system can previse extrapolation rules that allow for the missing data to be, well, imagined based on previous contexts—so that results are accurate.
- Networks: streaming analytics is already doing a great job at detecting and preventing security issues, from passive ones like erratic traffic and wiretapping to data modification attempts like cyberattacks, spoofing, VLAN jumping, phishing, or smurf attacks. Mitigation is often real-time and can involve automatic shutdowns, bans, and temporary suspension of services.
- Applications: from input validations and software tampering to session management and parameter manipulations, all can be kept at bay through continuous monitoring coupled with real-time, push-based actions led by business rules.

Self-healing: detecting + blocking + eliminating cause
We're getting better at designing patterns for fault-tolerant applications prevised with self-healing options for when internal processes fail. We're also fluent in moderating smaller software units capable themselves to self-heal, recuperate from failures, multiply,

or even self-destroy—but microservices + self-healing apps do not a self-healing system make.
At a system level, self-healing can be applied to all applications and services independently of their internal condition and is triggered not by individual issues but aspects like process failure- in other words, not how performant are the elements on their own, but how they work together.

Systems of intelligence can already monitor themselves and restore to the desired state once something goes awry, but humans are usually still needed to diagnose, fix the design, or eliminate a bug. What we're striving for are systems that not only find a way to keep going despite the malfunction-at hand (through hardware repairs, software bypasses, and so on) but also analyze the context of the failure and re-position themselves so future similar conditions are triggering action before the failure happens.

Self-perfecting: the system runs operational analytics continuously, exhaustively, and in real time to determine ways to optimize performance based on instructions as simple as: use less resources, produce more results, etc. Of course, the concepts of *less* and *more* are deeply subjective and entirely dependent on previous experiences and what we would call common sense (what's the minimum you can subtract for something to be an acceptable *less*?). However, randomization **can** be programmed, and easily added to the roster of routine monitoring operations.

Conclusion

The idea of random incursions into a system that works with the sole scope of assessing ways to make it better should not be that far-fetched. Random testings, corroborating non-related data (a Twitter flurry over a particular security-related term, for example) or dynamically moving the baselines for KPIs without human intervention are all examples of how a system-of-intelligence business can keep up with the connected consumer.

Moving from a Big Data, backward-looking analysis model to a forward-looking, actionable, **monitoring** world is well on the way. Systems of intelligence are, at least for now, the ultimate platform for capturing, analyzing, mediating, and sometimes executing all 3 types of digital interactions: human-to-human, human-to-machine, machine-to-machine.

And as they rise, here are the 7 SoI possibilities to follow:

1. **DevOps will become AllOps**
   First there were the Ops, then DevOps came about and officiated a much-celebrated symbiosis between network and application development functions. Deployments got easier and faster, and a new set of processes came about that made some companies rich, while some called out for NoOps in an attempt to further speed up the releases and deepen each professional's focus on their own area of expertise.
   In an SoI, roles are bound to shift a bit. Applications (both customer-facing and internal) will be developed and ran concurrently and continuously within the same unique system, which will **finally allow for operations to inform the business and vice versa**. The DevOps role will therefore switch from moderating existing processes to designing business logic rules applicable to the system-at-large.

2. **Security focus here to stay**
   With flexible reach and room to learn, intelligent systems will become a new kind of target for external attacks. With data security under monitoring and network and transactional security applications running on smart, reactive, adaptive platform, the weakness will increasingly come from the inside.

Internal identity management and specific access permissions design, for example, will have to be added to the roster of security protocol as an important aspect of organizational data strategy running on data from company devices and metadata created by processes and apps.

Gartner estimates that 15-20% of the security response functions currently performed by humans will be taken over by cognitive computing capabilities and prescriptive security analytics, so there you have it.

3. **Code won't change itself, so watch out for antipatterns**
   Much like DNA, code doesn't support directed mutations that can then take a path of their own; but development is a chaotic activity, and more often than not the implementations stray from the plan when context is altered by new constraints.

   Antipatterns are a sort of autoimmune illness one sometimes doesn't even know it has- and they can severely affect both the integrity and the efficiency of a system- unless, of course, it's a system of intelligence. Blobs, dead code, decomposition, boat anchors, dead ends, limited life-cycles and even mere obsolescence can be detected and addressed by systems built not only to protect or fix themselves, but also to simply improve automatically.

4. **Some money will be lost**
   If you're a business lead, be ready for some losses- not from increased operational costs (although there's a chance your R&D will go up), but from cannibalization.
   Rapid proliferation of business ideas will conduce to coupling products and services together in next-generation offerings that have to include AI in order for a business to remain competitive. These *intelligent automation services* use one or more AI technologies and are already becoming the basis of many core value propositions—taking value away from labor or licensing.

5. **The customer will rule it all**
   The accelerated adoption of AI is indeed transforming the way we do business, but the change is dictated not only by advancements in science, but the way the

consumer is choosing to interact with it—and what we have on our hands is a digital-native generation.
Currently, 63% of all applications are consumer-facing, and every new service and IoT device exponentially adds to the customer expectations. Will all else being equal, they can only be met through development-at-speed (of an application or service, for example) and the ability to dynamically adapt the offering to enhance the user experience in real time.

6. **There's no difference between IT and the business**
   With systems getting more complex and the market getting more competitive, the interdependence of hardware, software and applications constantly increases. For example, Forrester discovered that the number one issue affecting customer-facing engagements is faulty integration with the rest of the system.

   With customer needs evolving quickly (and customers themselves getting more fickle), business and applications are converging. As a result, the strategic role of IT—and the pressure on it to perform—increases. (Look at it this way: customer needs translate into business needs, and business needs can only be fulfilled through technology that needs to be developed or picked, ran, managed, and maintained. That's IT.)

   Programming, in other words, is here to stay while a few new IT functions are on the rise: machine training, writing code for bot interactions (particularly in customer service), and machine learning monitoring and guidance.

7. **You better have a cockpit**
   At the rate we're going (as per Gartner), the very near future holds **a live and interactive surrounding world, fed and operated by systems of intelligence that can continuously adapt to changing business conditions to maintain optimal operations and flawless customer experience.**

   In turn, the intelligence in these systems will come from both business transactions *and* operational insight. And in order to extract it, we'll need to be able to build, monitor, and reconfigure all of the system's components both independently and in the context of the system-at-large, concurrently, continuously, and in real time.

Infrastructure, operations, applications, and services on the same dashboard—sound overwhelming? It should be. There are plenty of technologies offering solutions to bits of the integer- development interfaces, analytics, monitoring, performance management tools like APMs. All do their part to resolve a hardware, software, or process problem, but that's not enough if they're not laced in together and handled in report to each other, and the whole system employing them.

What we need, it seems, is a central nervous system complete not only with a master-of-the-universe view (to collect and organize all input), but also dynamic design (so it's adaptive) and self-management capabilities (think autopilot).

The outcome (no need for a marketing cloud or APMs, analytics will change fabric, integrations will rule the function) may be controversial, expensive, hard to build, or difficult to manage.

And as far as intelligent-anything goes, it sounds about right.