The Tricorn framework (extract)

… to test and verify organizations' "security by design" and their capability for integration, but also to learn to exploit related opportunities & adjust systems/architectures to fit socio-technical environments!

Derived from "Because Linux is Law"

Author/candidate: Christopher Klooz

Presentation of 2023 (extract created 2025)

The Tricorn framework (extract)

This is an extract from a slide presentation (extract = full 6th chapter): the extract introduces the "tricorn" framework, which is derived from "Because Linux is law: mapping the socio-technical architecture around Linux and determining its institutional dynamics" and its data. The "tricorn" framework is derived from "Because Linux is Law", but the framework itself is not part of it. The presentation/extract does **not** contain a bibliography. The presentation is intended to be without speaker, with red text boxes adding/elaborating complementary notes.

The "tricorn" framework aims to provide a means to to test and verify organizations' "security by design" and their capability for integration, but also to learn to exploit related opportunities & adjust systems/architectures to fit socio-technical environments!

The extract presumes the reader to have read at least the **abstract** and **conclusion** of "Because Linux is Law"
→ The **full paper** including abstract and conclusion is **available online**:
https://github.com/py0xc3/PoliticalScienceAndGames/tree/master/linux-is-law

Full PDF direct link: *https://github.com/py0xc3/PoliticalScienceAndGames/raw/refs/heads/master/linux-is-law/RHUL.linux%20is%20law_Full.Paper.pdf*

Except the title slide and this slide, the extract's slides' content is equal to the original presentation of Dec. 2023 (formatting has been adjusted). The extract shall be coherent and conclusive without the other parts of the presentation.

6. Initial framework for testing, comparing & verifying

→ This is an initial framework to test, compare and/or verify existing systems and their architectures

→ It can reveal/show factual mixed states and the different manifests

→ It can illustrate the dynamics (through different manifests)

→ It can illustrate that organizations and architectures are never static, so are their delimitations! (It thus shows the *factual* organizations, which means organization manifests, that *occur* through *called* mixed states)

→ It can reveal yet unknown information flows (and thus unknown organization manifests: advantageous or disadvantageous)

→ It can reveal that some assumed information flows do not always exist when they are assumed (or exist in a different manner than assumed)

→ It can thus reveal security issues (with regards to all types of security)

→ yet, it is so far very generic and abstract, which limits use cases

→ *but it is a good point to start - and it can give the "facilitators & doers" a feeling of the dynamics that can be their friend or their foe!*

6. Initial framework for testing, comparing & verifying

→ This is an initial framework to test, compare and/or verify existing systems and their architectures

→ It can reveal factual mixed states and the different manifests

→ It can illustrate the dynamics

→ It can illustrate that organizations and architectures are never static, so are their delimitations

→ It can reveal unknown information flows

→ It can reveal that some assumed information flows do not always exist when they are assumed

→ It can thus reveal security issues (with regards to all types of security)

→ yet, it is so far very generic and abstract, which limits use cases

→ *but it is a good point to start - and it can give the "facilitators & doers" a feeling of the dynamics that can be either friend or their foe!*

This framework largely resembles my "follow the problem during its solving process" approach in "Linux is Law", which should be known and understood when reviewing the framework *(the previous parts of this presentation cover some of the major points)*; yet the framework also considers the "Linux is Law" conclusions, being a "generalized standardization" that does not dictate the focus on institutions and institutional dynamics in the actual analyses of the different stages; however, I still suggest to apply that focus for the same reasons as in "Linux is Law" (issue of pre-determined actors, and so on)

Any tool for evaluating types of competition can also be deployed (e.g., Porter's Five Forces plus the "complementary force" as additional force, Nash equilibria):

the means/approaches of Linux is Law remain my role model for the means to implement the framework.

6. Initial framework for testing, comparing & verifying

→ The system/architecture is tested/verified through “input”, “follow” the input on its “problem solving” way, and verify its “processing” stages

→ **Types of input** that might be tested and followed on the way through processing *(each input type should be tested with multiple/different informations of the very type) (see “Linux is Law” about the input types)*

1) intended input
2) unintended input
3) expected unintended input *(shall be always “pull”-based processing, never “push”!)*

The 2nd and the 3rd might evolve to:

4) detected unintended input

The 1st can be erroneously categorized as the 2nd/3rd & thus can evolve to:

5) wrongfully-categorized intended input

6. Initial framework for testing, comparing & verifying

→ At the same time, the places for “expected unintended input” (3) are maintained because they are likely to regularly get intended input (1) that then needs to be filtered, which follows the **rule**

*“prefer the **risk for getting unintended input** over **the risk to miss intended input**”*

→ e.g., 9 of 10 bug reports might be scam if the entry barrier to create a bug report is low, but the 10th might identify an issue that can cause a lot of harm to everyone and that would have been not identified if the entry barrier is too high

→ the rule aims to decrease the entry barrier to forward/pass information!

→ *Information flow/availability is everything!*

→ Every output is just the result of a physical, social or technical input *(at the most generic level: the only output without input is the big bang, which is the action to which everything else is just a reaction, or the reaction to the reaction*

6. Initial framework for testing, comparing & verifying

→ At the same time, the places for "expected unintended input" (3) are maintained because they are likely to regularly get intended input (1) that then needs to be filtered, which follows the **rule**

"*prefer the* ***risk for getting unintended input***
over ***the risk to miss intended input***"

→ e.g., 9 of 10 bug reports might be scam if the entry barrier to create a bug report is low, but the 10th might identify an issue that can cause a lot of damage to everyone and that would have been not identified if the entry barrier is too high

→ the rule aims to decrease the entry barrier without losing forward/pass information!

→ *Information flow/availability is everything!*

(an inevitable property for keeping the framework reproducible, consistent, logically-coherent and tenable, and without probabilistic assumptions)

→ Every output is just the result of a physical, social or technical input *(at the most generic level: the only output without input is the big bang, which is the action to which everything else is just a reaction, or the reaction to the reaction*

6. Initial framework for testing, comparing & verifying

→ At the same time, the places for "expected unintended input" (3) are maintained because they are likely to regularly get intended input (1) that then needs to be filtered, which follows the **rule**

"*prefer* the ***risk for getting unintended input***
over ***the risk to miss intended input***"

→ e.g., 9 of 10 bug reports might be scam if the entry barrier to create a bug report is low, but 1 of 10 might identify an issue that can cause a lot of harm for everyone and that would have been not identified if the entry barrier is too high

→ the aim is to decrease the entropy for valuable information!

→ *Information flow/availability is everything!*

→ Every output is just the result of a physical, social or technical input (*at the most generic level: the only output is the input is the big bang and this is the action to which everything else is just reaction, or the reaction to the reaction*

A comparable principle has proven competitive in wars (which is the most generic competition, taking place at the "state of nature" without abstractions) and became a major rule in warfare in armies:

Roughly translated, "impact over cover"

→the principle here could be argued to be an implementation of the army one (…every agency comes down to competition…)

(yet, a passive possibility for input/impact does not automatically imply active use/impact!)

6. Initial framework for testing, comparing & verifying

→ The first stage at each “point of processing” is **quantitative** (true/false):

→ test for the following properties and create incentives to **create** them if not achieved **by default**, then test again:

1) No single point of failure
2) Enforced intended input (“pass what is explicitly allowed”)
   OR in case of “expected unintended input”:
   any place that does not enforce intended input has to enforce ”trusted” “pull”-based processing *(so, the “puller” is trusted)*
   (through the 1st point, the 2d might evolve to more than one “pull” before classifying input to be “intended”, OR one “pull” has to be reviewed/confirmed by a multiple entities)

6. Initial framework for testing, comparing & verifying

→ The first stage at each “point of processing” is quantitative (true/false):

→ test for the following properties and create incentives to create them if not achieved by default, then test again:

3) Keep original data/information/request retained as it is the core of the actual “problem processing”: it is the **immutable** center of the debate and it constitutes the actual problem. Retaining this avoids **mutations** of “problem processing” & thus risks of “**mutated solutions**” that **no longer fit the original need.**

→ Also, the original reasoning / “questions” of the problem origin can reveal that the actual problem is different than expected, e.g., due to the asker’s lack of knowledge, derived from their questioning/reasoning

→ Also, this mitigates **mutations/splits of interests** in between *point of problem* & *point of solution*, e.g., Nash equilibria in internal competition

6. Initial framework for testing, comparing & verifying

→ The first stage at each "point of processing" is quantitative (true/false):

→ test for the following properties and create incentives to create them if not achieved by default, then test again:

3) Keep original data/information/request retained as it is the core of the actual "problem processing": it is the **immutable** center of the debate and it constitutes the actual problem. Retaining this avoids **mutations** of "problem processing" & thus risks of "**mutated solutions**" that **no longer fit the original need.**

→ Also, the original reasoning / "questions" of the problem origin can reveal that the actual problem is different than expected, e.g., due to the asker's lack of knowledge, derived from their questioning/reasoning

→ Also, this mitigates **mutations/splits of interests** in between *point of problem* & *point of solution*, e.g., Nash equilibria in internal competition

This could be argued to be a type of "distortion of competition" (to the disadvantage of the dynamics that cause the affected and related manifests: in short, to the disadvantage of the "agency")

6. Initial framework for testing, comparing & verifying

→ The first stage at each “point of processing” is quantitative (true/false):

→ test for the following properties and create incentives to create them if not achieved by default, then test again:

…

…

…

→ if any alteration takes place at any of the properties of this stage, repeat the whole stage again until all properties are unaltered within one iteration

6. Initial framework for testing, comparing & verifying

→ The second stage at each "point of processing" is **qualitative** (determine if intended minimum is fulfilled, and if possible improve):

→ test for the following properties and create incentives to **create or improve** them, then test again:

1) simple → less vulnerable and easier to understand & predict
2) outreach/coverage for/of knowledge/processors (incl. forwarders)
3) competitive → *reliability (which can be equal to replaceable; relation to 1st stage), efficiency, flexibility/adaptivity*

→ Negative correlations of property adjustments (in short, compromises / trade-offs) should be investigated!

→ If any alteration takes place at any of the properties of this stage, repeat the whole stage again until all properties are unaltered within one iteration

6. Initial framework for testing, comparing & verifying

→ The second stage at each "point of processing" is **qualitative** (determine if intended minimum is fulfilled, and if possible improve):

→ test for the following properties and create incentives to **create or improve** them, then test again:

1) simple → less vulnerable and easier to understand & predict
2) **outreach/coverage for/of knowledge/processors**

→ "understanding", which Fogg (2009) lacked to consider, has to be organizational, not individual: everyone has to understand **when what** to trigger to achieve that the *point of knowledge* (or, *innovation*) gets connected to the *point of need* (in the end, achieve an effective supply & demand).

→ Then, if this is "perfectly achieved", understanding is only relevant from an organizational perspective because it would imply that a point of "need & knowledge"-connecting manifest will occur if a related point of knowledge exists in the affected "agency of dynamics".

6. Initial framework for testing, comparing & verifying

→ The second stage at each "point of processing" is **qualitative** (determine if intended minimum is fulfilled, and if possible improve):

→ test for the following properties and create incentives to **create or improve** them, then test again:

1) simple → less vulnerable and easier to understand & predict
2) **outreach/coverage for/of knowledge/processors**

→ introducing points of *expected unintended* input (and incentives to use/exploit them) can facilitate many triggers/knowledge-transfer for those who do not know **what/where** to trigger, whereas those who know what they can "solve" will know what to pull *(places of input that roughly fit a predictable direction from the enquirer perspective)*, alternatively it is likely that **any reviewer** will know **where** to **forward** it next *(if the place of expected unintended input is at least chosen in the "roughly right direction", this is very likely! … and in any case, no competent attacker would rely on the low likelihood).* ***Redundancy is no problem here!***

6. Initial framework for testing, comparing & verifying

→ The second stage at each “point of processing” is **qualitative** (determine if intended minimum is fulfilled, and if possible improve):

→ test for the following properties and create incentives to **create or improve** them, then test again:

1) simple → less vulnerable and easier to understand & predict
2) **<u>outreach</u>/<u>coverage</u> for/of <u>knowledge</u>/<u>processors</u>**

→ places of *expected unintended* input do not need to be dedicated: e.g., many/most points of input at Linux-related communities are designed for *expected unintended* input although formally regulated for *intended* input: many *working groups*/*teams* do not reject what is not intended for them but encourage proceeding and help to forward (*reward* all ends of communication for any contribution)

→ they exploit that humans are more capable to process the question “is this intended for me or not?” than machines (social institution)

6. Initial framework for testing, comparing & verifying

Yet, not any place can "just be made" a place of *expected unintended* input!
→in some cases, it is not competitive because allowing any input would end up in too much input that can no longer be processed (practically a self-made "denial of service") or because the place of input can for some reason not be tailored to be able to process unintended input and thus is not able to expect it
→e.g., in many kernel mailing lists, unintended input is simply ignored, which should not be taken seriously but interpreted correspondingly
→e.g., some places of input should be not public for many possible reasons

→in person-centric organizations, privacy is less an issue in this respect since the organization is centered around people, which are not secret themselves
→such places of input might limit themselves to "intended input" and forward to *explicitly exclude* leaks

→yet, not any place does explicitly possess the capability for this limitation and thus some coverage/outreach might be not advantageous to be available from everywhere *(formalization of this issue tbd)*

2) outreach/coverage for/of knowledge/processors

→ places of *expected unintended* input do not need to be dedicated:
e.g., many/most points of input at Linux-related communities are designed for *expected unintended* input although formally regulated for *intended* input. many *working groups/teams* do not reject what is not intended for them but encourage providing and help to find (reward) all ends of communication (e.g. any contribution)
→ exploit that humans are more capable to process the question "is this intended for me or not?" than machines (social institution)

6. Initial framework for testing, comparing & verifying

→ The second stage at each "point of processing" is **qualitative** (determine if intended minimum is fulfilled, and if possible improve):

→ test for the following properties and create incentives to **create or improve** them, then test again:

1) simple → less vulnerable and easier to understand & predict
2) **outreach/coverage for/of knowledge/processors**

→ goal: facilitate requests, exchange & massive review from many perspectives of the *request* (it is likely that it is triggered to someone who can solve it, or facilitate a debate of what is the best way and who can do it best) → might also foster new knowledge/capabilities/outreach/coverage in the organizational architecture

→ foster rewards & no punishment for (preventive) *requests* (to solve or improve) & for reviewing/suggesting, and outreaching wherever feasible → competitive & collaborative culture (*Saxenian:* "collaborative competition")

6. Initial framework for testing, comparing & verifying

→ The second stage at each "point of processing" is **qualitative** (determine if intended minimum is fulfilled, and if possible improve):

→ test for the following properties and create incentives to **create or improve** them, then test again:

1) simple → less vulnerable and easier to understand & predict

2) **outreach/coverage for/of knowledge/processors**

→ goal: facilitate requests, exchange & massive review from many perspectives of the *request* (it is likely that it is triggered to someone who can solve it, or facilitate a debate of what is the best way and who can do it best) → might also foster new knowledge/capabilities/outreach/coverage in the organizational architecture

→ foster rewards & no punishment for (preventive) *requests* (to solve or improve) & for reviewing/suggesting, and outreaching wherever feasible → competitive & collaborative culture (*Saxenian:* "collaborative competition")

→the bigger the factual architecture/system (so, the size and amount of occurring/possible manifests from the related agency of dynamics), the better

(→which implies: decrease in exercising privacy = less confidentiality = larger size of system and thus of available knowledge → much culture, too!)

6. Initial framework for testing, comparing & verifying

→ The second stage at each "point of processing" is **qualitative** (determine if intended minimum is fulfilled, and if possible improve):

→ test for the following properties and create incentives to **create or improve** them, then test again:

1) simple → less vulnerable and easier to understand & predict
2) **outreach/coverage for/of knowledge/processors**

→ **critical issue today** in the transition from social organizations that use technology **to** socio-technical organizations that integrate technology *on all levels*: humans derive solutions for technical & socio-technical problems from social experience (tech. **pattern** that seem to resemble social pattern)

→ even if pattern/symptoms seem to be comparable, causality/origin is different and thus the appropriate solution

→ *today, much more critical manifests that are the outcome of wrongfully-determined pattern in human minds – and mostly, no "outreach-increasing" type of request for solution occurs at all)*

6. Initial framework for testing, comparing & verifying

2) outreach/coverage for/of knowledge/processors

→ **critical issue today *(continuing)***:

→ yet, people are able to understand by default if the immediately experienced problem is **technical** (e.g., problem shown on screen=technology) or **social**

→ teach them causalities and differences in between social and technical

→ make the **first pattern** that is **triggered** in **their minds** to be "tech or social causalities?" (triggering starts in human minds, not organizations)

→ teach with issues they know (how to understand a problem and thus trigger to solve it "securely", not dedicated security stuff that ignores underlying causalities and that *produces* "uncalled, actual-problem-unrelated uncompetitive trigger" anyway); tell them: why does this happen that way? How does that work? Why? Why can I not just proceed working here? Why does Firefox reject to open that page in this situation? What is the origin of when it works and when not? Why can a bug in the WiFi driver cause the graphics driver to break? What and thus who is related to this type of problem? How could I improve & trigger this to make my way of work better? ***(just some general reasonings without tailoring them for a specific audience)***

6. Initial framework for testing, comparing & verifying

2) **outreach/coverage for/of knowledge/processors**

→ **critical issue today *(continuing)***:

→ "**understanding**" fosters Fogg's (2009) "**motivation**" & **vice versa**

→ I conducted such a program with non-tech people with good outcomes (2017):

people get motivated to delve deeper (especially when related to privately-relevant stuff) because it is interesting to get understanding of what happens, e.g., when Facebook is not available in some situations, understand what is related to what & how to derive solutions and whom to ask, increased capability to identify that something is wrong and someone needs to be asked even if the very problem is not taught in advance, understanding whom to ask, even more sophisticated than "start with an IT guy", predict when the issue might appear again in order to better plan for that (and to ask related people in advance)

→ understand different causalities of technical & social problems

→ our minds are problem solvers, and our minds even biologically reward us if we solve problems or contribute to problem solving *(but the biological reward can only happen if people's minds understand that they contribute)*

6. Initial framework for testing, comparing & verifying

2) **outreach/coverage** of **knowledge/processors**

→ **critical issue today**

→ "**understanding**" fosters Fogg's (2009) "**motivation**" & vice versa

→ I conducted such a program with non-tech people with good outcomes (2017):
people get motivated to delve deeper (especially when related to privately-relevant stuff) because it is interesting to get understanding of what happens e.g. when Facebook is not available in some situations, understand what is related to what & how to derive solutions and whom to ask, increased capability to identify that something is wrong & need someone needs to be asked when the very problem is not taught in advance, understanding whom to ask, even more sophisticated than "stay with an IT guy", prevent when the issue might appear again in order to have plan for that (need to ask related people in advance)

→ understand different causalities of technical & social problems

→ our minds are problem-solvers, and our minds even biologically reward us if we solve problems or contribute to problem solving *(but the biological reward can only happen if people's minds understand that they contribute)*

→ my subjective perception of the *rough* experiments:
understanding & motivation correlate positively with each other and with constructive problem solving and thus with absence of unintended/unforeseen security breaches or related issues
→ people do not need to understand each causality in between the kernel and the web browser, but only the generic "reasoning of technical causalities" and how they evolve in order to differentiate them from social causalities (and thus roughly differentiate causal relations and developments of the machine and humans) → create a proper "initial trigger" to get the problem pushed, and to push it into the right direction!
... Everything else evolves over time intuitively ...

6. Initial framework for testing, comparing & verifying

2) outreach/coverage for/of knowledge/processors

→ **critical issue today *(continuing)***:

→ "continuous socio-technical evolvement" in dynamic/changing environments has to be an integrated part of the system: if the system stalls to develop constructively, check all stages to identify the halt *(e.g., divergent interests? No undetstanding at some point?)*

→ dynamic evolvement has to be intuitive and integrated into the system

→ military rule-of-thumb: **every soldier is a recon** *who might trigger incentives/information/questions/requests that contribute to intelligence!*

→ because war is just competition, this rule applies to any system!

→ only the system has perfect information about the system and its immediate environment – but no individual within *(which makes enforced adjustments / "distortions of competition" within always to a gamble to some extent!)*

→ the more changes happen to the environment and the more dynamic it is, the more devastating are tries to enforce *statics* & *imperfect decision-making* in the *agency of dynamics* and its *system manifests* and the more subliminal & unnoticed their actual developments will be ...

6. Initial framework for testing, comparing & verifying

2) outreach/coverage for/of knowledge/processors

→ **critical issue today *(continuing)***:

→ "continuous socio-technical evolvement" in dynamic/changing environments has to be an integrated part of the system: if the system stalls to develop constructively, check all stages to identify the halt *(e.g., divergent interests? No undetstanding at some point?)*

→ dynamic evolvement has to be intuitive and integrated into the system

→ military rule-of-thumb: **every soldier is a recon** *who might trigger incentives/information/questions/requests that contribute to intelligence!*

→ because war is just competition this rule applies to any system!

→ only the system has perfect information about the system and its immediate environment *(which makes enforced adjustments "distortions of competition" within always to a gamble to some extent)*

… and therefore, the more any point of input & processing is forced to balance *locally* the *distortions of competition* into unforeseeable distorted directions in order to locally remain competitive!

→ the more changes happen to the environment and the more dynamic it is, the more devastating are tries to enforce *statics* & *imperfect decision-making* in the *agency of dynamics* and its *system manifests* and the more subliminal & unnoticed their actual developments will be ...

6. Initial framework for testing, comparing & verifying

2) <u>outreach</u>/<u>coverage</u> for/of <u>knowledge</u>/<u>processors</u>

→ **critical issue today** ***(continuing)***:

→ **dedicating security** in general just **blurs facts** → it is just about problem solving:

→ **security** is just **getting permanently what is expected/intended**

→ it is a means to compare and trade-off different properties/intentions (different types/manifests of security):
security to have competitive advantage, to not loose customer data, to have efficient and flexible but reliable and predictable operations, and so on)

→ "when does who outreach/cover which direction in which situation (or with which input)" **gives a lot of indication**: first understand it yourself, then make the system understand… then, together improve problem solving capabilities! This is secure...

→ **steer and give incentives**, then **observe and adjust** - do **not** order because competition will find its way! **Suppressing** it only ensures that we **do not know when it will re-manifest where in which way**: **insecurity** is the outcome!

6. Initial framework for testing, computing & verifying

2) Outreach/coverage for/of knowledge processors

→ **critical issue today** *(continuing)*:

→ "continuous socio-technical evolvement" has to be integrated part of the system: if the system stalls to develop constructively, check all stages to identify where to break the stall (e.g., *divergent interests? No understanding at some point?*)

→ dynamic evolvement has to be intuitive and integrated into the system

→ military rule-of-thumb: *every soldier is a sensor who might trigger incentives/information/questions that contribute to intelligence!*

→ because war is just competition, this rule applies to any system!

→ the more changes happen to the environment and the more dynamic it is, the more demanding are tries to enforce "static" systems/architectures and the more subliminal & unnoticed their actual developments will be, and likewise the more silly place of input and of processing is needed to dynamically & intuitively adjust & forward properly!

→ the continuous evolvement process has to be rationalized and steered, not its outcome or the properties!

The latter would assume that any managing entity has perfect information of the organization! This is not realistic...

Instead, steer, exploit and trust competitive dynamics that always are the outcome of perfect information and that integrate any related intelligence:

→ it is the incentives to achieve the desired competition that are critical;

the desired competition is always a collaborative one (*collaboration actually makes and delimits the factual system* and the "*agency of dynamics*" that we perceive in a "simplified and distorted" way as the "*organization delimited from other organizations* throughout *manifests*")!

→ alternatively, the *organization* can be seen definition-dependent either as the coverage of all possible inputs, or there are many factual organizations with each input triggering another organization, each with a unique agency!

→ considering the "agency of dynamics" (or, "agency of collaborative dynamics") as organization seems most practical so far, but it is already a *summary* that should be derived only for the result of analyses, not as interim-result that is itself used as data for further analyses!

6. Initial framework for testing, comparing & verifying

2) outreach/coverage for/of knowledge/processors

→ **critical issue today** ***(continuing)***:

→ "continuous socio-technical evolvement" has to be integrated part of the system: if the system stalls to develop constructively check all stages to identify where to break the stall *(e.g. changing interests? No understanding at some point?)*

→ dynamic evolvement has to be intuitive and integrated into the system

→ military rule-of-thumb: **every soldier is a recon** *who might trigger incentives/information/questions that contribute to intelligence!*

→ because war is just competition, this rule applies to any system!

→ the more changes happen to the environment and the more dynamic it is, the more devastating are tries to enforce "statics" in systems/architectures and the more subliminal & unnoticed their actual developments will be, and likewise the more any place of input and of processing is needed to dynamically & intuitively adjust & forward properly!

→the outreach / coverage properties in the 2nd stage might be most related to Saxenian's research: the outreach / coverage is the outcome of an effective and efficient collaborative competition

→the framework might achieve much improvement at this place when analyzed against Saxenian's output & conclusions

However, all properties of all stages might profit from each other's improvement

6. Initial framework for testing, comparing & verifying

→ The third stage at each "point of processing" can be **quantitative** (true / false) and/or **qualitative** (determine min., and if possible improve):

→ test for the following property and create incentives to **create or improve** them, then test again:

1) **Sustainability** of ***hard*** and ***soft*** **dependencies**: sustainability is a generic term, which can be related to, e.g., *environmental sustainability* but does not have to.

→ In the end, this measures the **security of competitive advantage** of dependencies, and **only if** *security of competitive advantage* is achieved, the security of CIA* will be considered by any entity (→ ***if security / CIA* is not competitive, the organization / architecture will intuitively get rid of it: formally or informally…***) *[* security of **C**onfidentiality, **I**ntegrity and **A**vailability]*

(enforcing = distortion of competition, unpredictable outcome, etc: as elaborated earlier)

6. Initial framework for testing, comparing & verifying

→ The third stage at each "point of processing" can be **quantitative** (true / false) and/or **qualitative** (determine min., and if possible improve):

→ test for the following property and create incentives to **create or improve** them, then test again:

1) **Sustainability** of *hard* and *soft* **dependencies**: sustainability is a generic term, which can be related to, e.g., *environmental sustainability* but does not have to.

→ In the end, this measures the **security of competitive advantage** of dependencies, and **only if** *security of competitive advantage* is achieved, the security of CIA* will be considered by any entity ***(→ if security / CIA* is not competitive, the organization / architecture will intuitively get rid of it: formally or informally…)*** *[* security of Confidentiality, Integrity and Availability]*

This might have a relation to the decrease in privacy needs and thus decrease in confidentiality needs: it is not competitive for many if not most entities/people in the current environment: enforcing many types of privacy ain't easy, but a society can adjust its informal institutions to make it obsolete in some fields)

6. Initial framework for testing, comparing & verifying

→ The third stage at each "point of processing" can be **quantitative** (true / false) and/or **qualitative** (determine min., and if possible improve):

→ test for the following property and create incentives to **create or improve** them, then test again:

1) **Sustainability** of ***hard*** and ***soft*** **dependencies**: sustainability is a generic term, which can be related to, e.g., *environmental sustainability* but does not have to.

A) **Security of competitive advantage:**

reliability *(which can be equal to* ***replaceable****; relation to 1st stage)*, **efficiency, flexibility/adaptivity**

→ can be fostered by ***simplicity*** *(relation to 2nd stage: but it is potentially neither competitive nor possible to try to measure simplicity of dependencies)*

6. Initial framework for testing, comparing & verifying

→ The third stage at each "point of processing" can be **quantitative** (true / false) and/or **qualitative** (determine min., and if possible improve):

→ test for the following property and create incentives to **create or improve** them, then test again:

1) **Sustainability** of ***hard*** and ***soft*** **dependencies**: sustainability is a generic term, which can be related to, e.g., *environmental sustainability* but does not have to.

(relation to 1st stage)

B) Avoid any dependency to be able to act as

- single point of failure **- place of "unintended input"**

→ is there a place of "expected unintended" input the dependency has to pass before its output can affect the system?

- if **no**, actively and explicitly ensure "intended" or "expected unintended" input **from** the dependency
- if **yes**, it is "expected unintended" input anyway

6. Initial framework for testing, comparing & verifying

→ The third stage at each "point of processing" can be **quantitative** (true / false) and/or **qualitative** (determine if and if possible improve):
→ test for the following property and create incentives to **create** or **improve** them, then test again:

1) **Sustainability** of ***hard*** and ***soft* dependencies** (sustainability is a generic term, which can be related to, e.g., *environmental sustainability* but does not have to):

(relation to 1st stage)

B) Avoid any dependency to be able to act as
- single point of failure **- place of "unintended input"**
→ is there a place of "expected unintended" input the dependency has to pass before its output can affect the system?
- if **no**, actively and explicitly ensure "intended" or "expected unintended" input **from** the dependency
- if **yes**, it is "expected unintended" input anyway

Coverage ain't useful here: its beyond effective control, and coverage/outreach has to be **within** the system, whatever the system's very mixed state is: coverage/outreach already includes "passing points" towards solutions/knowledge (which become part of the mixed state) → thus the dependency is by definition not part of the coverage, but coverage can depend on dependencies that act as facilitator (but, e.g., dependency providers can be part of coverage!)

6. Initial framework for testing, comparing & verifying

→ The third stage at each "point of processing" can be **quantitative** (true / false) and/or **qualitative** (determine min., and if possible improve):

→ test for the following property and create incentives to **create or improve** them, then test again:

1) **Sustainability** of ***hard*** and ***soft*** **dependencies**: sustainability is a generic term, which can be related to, e.g., *environmental sustainability* but does not have to.

→ **Security of competitive advantage:**

…

…

…

→ if any alteration takes place at any of the properties of this stage, repeat the whole stage again until all properties are unaltered within one iteration

6. Initial framework for testing, comparing & verifying

→ **Important relations among the three stages**: e.g., dependencies might **break a part of the system**, but if the **affected part of the system itself** can be **easily replaced/balanced**, reliability issues of the dependencies might be **neglected**, and vice versa: absence of single points of failure can be achieved on multiple levels of problem solving, but it has to be ensured at any *(… in the dynamics, not only at a given manifest!)*.

→ … the same for **enforcing intended input** or "pull"-based processing: "expected unintended" inputs do not have to be determined by the system to be "unintended" or "intended" immediately at the first processing, but it has to be clear as long as they are potentially "unintended" and thus processed correspondingly.

→ Therefore, points/parts/places & levels have to be **set in contrast** to each other!

6. Initial framework for testing, comparing & verifying

→ **Important relations among the three stages** (e.g., dependencies) might cause a **break of a part of the system**, but if the **affected part of the system** can be **often easily replaced**, reliability issues of the dependencies might be **neglected**, and vice versa; absence of single points of failure can be achieved on multiple stages/levels of problem solving, but in every manifest/fixed state, it has to be ensured at any

→ the same for **enforcing intended input** or "pull"-based processing: "expected unintended" inputs do not have to be determined by the system to be "unintended" or "intended" immediately at the first stage, but it has to be clear as long as they are potentially "unintended" and thus processed correspondingly. The actual "pull" that accepts inputs from "expected unintended" needs

→ stages have to be **set in contrast** to each other!

Keep in mind that it is about socio-technical inputs, and inputs have to be considered and differentiated correspondingly:
Your bugzilla service might be considered to cause "expected unintended" input from users in its tickets, but the availability of the bugzilla and its technical/back-end inputs can still be "intended" input and cause problems if not acting as "intended":
It can still cause problems if someone can, e.g., hack the bugzilla service and manipulate bug tickets (e.g., to delete/alter reports that reveal security vulnerabilities the attacker wants to exploit)

6. Initial framework for testing, comparing & verifying

→ **If a risk to the system is accepted,** test the vulnerable system part to be itself separated, e.g., by "expected unintended" input to critical system parts

→ keep in mind that the vulnerable system parts might have **different information flows** and **outreaches** in different dynamics / mixed states

→ dynamics / mixed states can be also limited by "unexpected unintended" input (don't forget: **people are the constants/statics**!)

→ there is not just the "let everything in" "expected unintended input"

→ this adds complexity through "input type iterations": "intended expected intended" input

Too complex and thus infeasible to achieve:

Do not accept risks! Just accept dynamics that manage risks. Achieve advantageous collaborative competition and trust it: if it fails in this respect, the issue is much bigger anyway and it is anyway arbitrary if static enforcements make a difference in the very mixed state

6. Initial framework for testing, comparing & verifying

→ At the end, repeat all stages until in at least one iteration all stages have been processed without any alteration in any of its properties

→ In any case, repeat iterations at a later time again to compare and to measure developments over time (might reveal previously-unknown information / unknown development of dynamics), and to identify changes in the architecture / system that are maybe not yet known OR that are expected/intended (the latter means to verify success of alterations & thus measure the environment that also can be provided with incentives)

→ changes in the system can affect (and thus change) the environment, whereas the environment then makes new changes within the system necessary: circle of reciprocal influences (=dynamics)

→ Obviously, interrelations of the stages might cause the third stage to **partly** resemble a "meta stage" (with nonhazardous exceptions) for 1st and 2nd stages *(in an overall system, the whole system/architecture can be **depicted as a dependency** for something: things without purpose do not occur in nature!*

6. Initial framework for testing, comparing & verifying

→ Repeat all stages until in at least one iteration all stages have been processed without any alteration in any of its properties

→ Repeat iterations at a later time again to compare and to measure developments, and to identify changes in the architecture / system that are maybe not yet known OR that are expected/intended (=verify success of alterations & thus measure the environment that can be changed as well)

→ changes in the system can affect (and thus change) the environment, whereas the environment then makes new changes within the system necessary: circle of reciprocal influences

→ obviously, interrelations of the stages might cause the third stage to loosely resemble a “meta stage” for 1st and 2nd stages *(in an overall system, the whole system/architecture can be depicted as a dependency for something)*

The analyses of different inputs (different input types) can be compared against each other!

Differences of manifests (e.g., of coverage/outreach) of different inputs of the same or might be valuable indication!