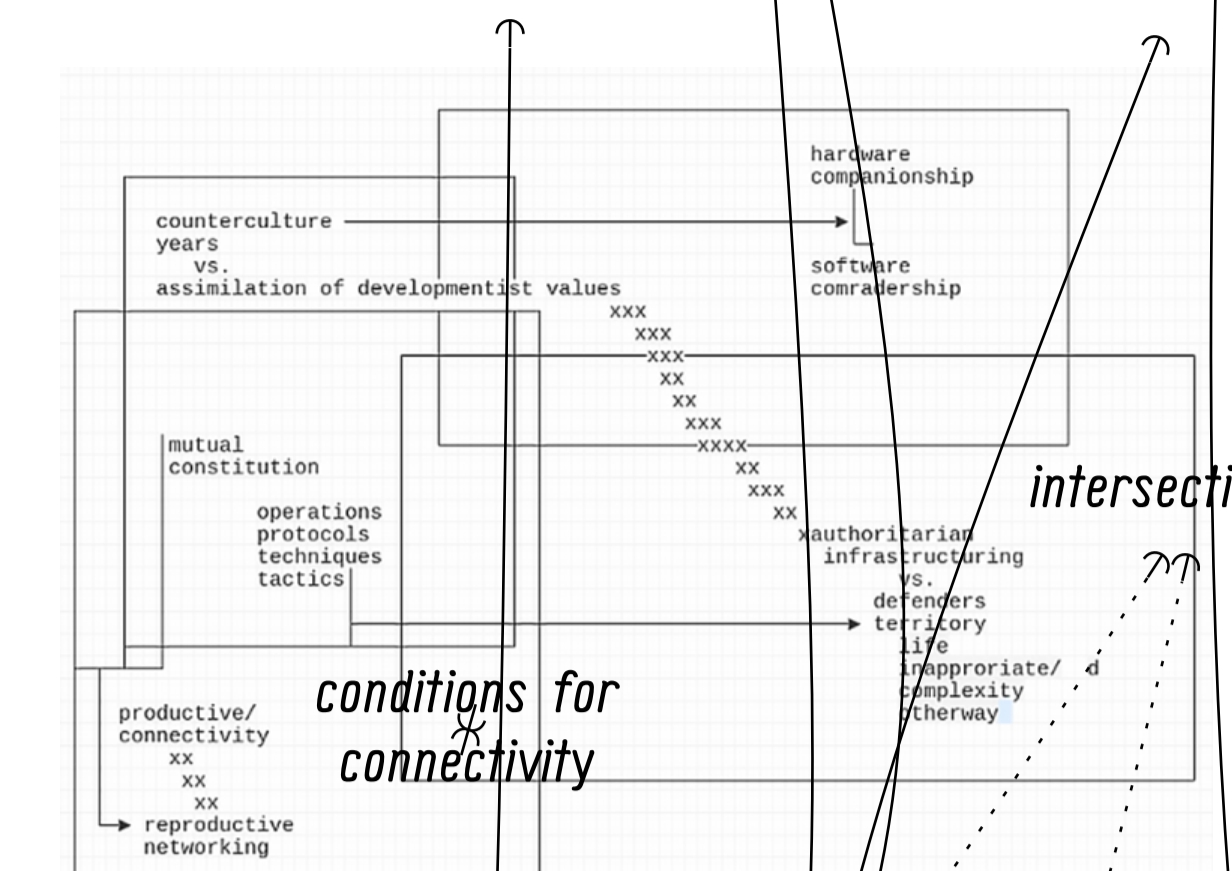


self-sovereignty
technological sovereignty
autonomy
interdependency
strategic autonomy
relational sovereignty
solidarity

Warning: the bounding boxes of some nodes were calling back to straight line edges failed at node 0011
slope: matrix:212: edgeGraph: Assertion: top-cells[] failed. Aborted (core dumped)



interoperability
operations
infrastructure
connectivity
software companionship
solidarity

radical allyship
NOT ALLIES
Abolishing the ally industrial complex, an indigenous perspective

insurgent intersectionality
trans-sectionality
maintenance

institutional manoeuvres
infrastructural free manoeuvres
smooth-flow

entanglement
seamfulness
burrowing

friction
federation
horizontal integration
horizontal integration

centralization
topological transformation

"In the wormhole the worm creates an infrastructure to hold itself in the world: the hole is the worm, but only as it moves." (On the inconvenience of other people, Lauren Berlant)

"We might imagine re-creating a multiplicity of processes, such as the kinds earthworms meet in while helping to make compost or otherwise being busy at work and at play: turning the soil over and over - ingesting and excreting it, tunnelling through it, burrowing, all means of aerating the soil, allowing oxygen in, opening it up and breathing new life into it" (Distinguishing Différance: Caring Together, Kate Raworth)

Myocellum, circular cascading restructure (decentralized but hierarchical)

territory
bundled beings
unbundling
multi-stationary
time sharing
digital separation
outsourcing
ambiguity

state sovereignty
reproductive networking
productive connectivity
no borders
boundaries do not still
state
control

so-and-sovereignty
queer angles
flow management
free flows

institutional exhaustion
affective infrastructure
Indeterminate precarity

ambivalence
Ambivalent precarity
affect

institutional detachment
institutional exhaustion
affective infrastructure
Indeterminate precarity

organized abandonment
so-called austerity
institutional ingratitude
dissolution

the cyst model
endosymbiosis
trembling
no

horizontal integration
horizontal integration
horizontal integration

horizontal integration
horizontal integration
horizontal integration

horizontal integration
horizontal integration
horizontal integration

horizontal integration
horizontal integration
horizontal integration

"The 3300 is a time sharing mini-computer system for only 1/4 the cost of subscription services or other in-house time sharing systems. That, in itself, is very interesting!"

by using an architectural style that discourages tight coupling, developers are able to establish cohesive boundaries among components, keeping the cost of coordinating software components to a minimum. Tight coupling also makes it easier to distribute computing resources and allocate responsibilities to different components of the system.

Virtualization is a process that allows a computer to be hardware resource. It is a logically separated virtual machine within an allocated resource. Each virtualized computer has its own memory, processing power, and storage.

Virtualization is a process that allows a computer to be hardware resource. It is a logically separated virtual machine within an allocated resource. Each virtualized computer has its own memory, processing power, and storage.

Precarity is ambivalent, because we are always dependent on other people from the beginning but other people can also harm us, so we need an understanding of ethics to cope with this ambivalence (Judith Butler)

Precarity is ambivalent, because we are always dependent on other people from the beginning but other people can also harm us, so we need an understanding of ethics to cope with this ambivalence (Judith Butler)

Precarity is ambivalent, because we are always dependent on other people from the beginning but other people can also harm us, so we need an understanding of ethics to cope with this ambivalence (Judith Butler)

Precarity is ambivalent, because we are always dependent on other people from the beginning but other people can also harm us, so we need an understanding of ethics to cope with this ambivalence (Judith Butler)

Precarity is ambivalent, because we are always dependent on other people from the beginning but other people can also harm us, so we need an understanding of ethics to cope with this ambivalence (Judith Butler)

Precarity is ambivalent, because we are always dependent on other people from the beginning but other people can also harm us, so we need an understanding of ethics to cope with this ambivalence (Judith Butler)

Precarity is ambivalent, because we are always dependent on other people from the beginning but other people can also harm us, so we need an understanding of ethics to cope with this ambivalence (Judith Butler)

Precarity is ambivalent, because we are always dependent on other people from the beginning but other people can also harm us, so we need an understanding of ethics to cope with this ambivalence (Judith Butler)

"(It) is not about a relationship between the tangible and the intangible, it is about the fixed duration of a relationship of trust, and a way of structuring the future in terms of liability and responsibility" (The durability of software, Ericsson & Kirby)

By using an architectural style that discourages tight coupling, developers are able to establish cohesive boundaries among components, keeping the cost of coordinating software components to a minimum. Tight coupling also makes it easier to distribute computing resources and allocate responsibilities to different components of the system.

By using an architectural style that discourages tight coupling, developers are able to establish cohesive boundaries among components, keeping the cost of coordinating software components to a minimum. Tight coupling also makes it easier to distribute computing resources and allocate responsibilities to different components of the system.

By using an architectural style that discourages tight coupling, developers are able to establish cohesive boundaries among components, keeping the cost of coordinating software components to a minimum. Tight coupling also makes it easier to distribute computing resources and allocate responsibilities to different components of the system.

By using an architectural style that discourages tight coupling, developers are able to establish cohesive boundaries among components, keeping the cost of coordinating software components to a minimum. Tight coupling also makes it easier to distribute computing resources and allocate responsibilities to different components of the system.

By using an architectural style that discourages tight coupling, developers are able to establish cohesive boundaries among components, keeping the cost of coordinating software components to a minimum. Tight coupling also makes it easier to distribute computing resources and allocate responsibilities to different components of the system.

By using an architectural style that discourages tight coupling, developers are able to establish cohesive boundaries among components, keeping the cost of coordinating software components to a minimum. Tight coupling also makes it easier to distribute computing resources and allocate responsibilities to different components of the system.

By using an architectural style that discourages tight coupling, developers are able to establish cohesive boundaries among components, keeping the cost of coordinating software components to a minimum. Tight coupling also makes it easier to distribute computing resources and allocate responsibilities to different components of the system.

By using an architectural style that discourages tight coupling, developers are able to establish cohesive boundaries among components, keeping the cost of coordinating software components to a minimum. Tight coupling also makes it easier to distribute computing resources and allocate responsibilities to different components of the system.

By using an architectural style that discourages tight coupling, developers are able to establish cohesive boundaries among components, keeping the cost of coordinating software components to a minimum. Tight coupling also makes it easier to distribute computing resources and allocate responsibilities to different components of the system.

By using an architectural style that discourages tight coupling, developers are able to establish cohesive boundaries among components, keeping the cost of coordinating software components to a minimum. Tight coupling also makes it easier to distribute computing resources and allocate responsibilities to different components of the system.

By using an architectural style that discourages tight coupling, developers are able to establish cohesive boundaries among components, keeping the cost of coordinating software components to a minimum. Tight coupling also makes it easier to distribute computing resources and allocate responsibilities to different components of the system.

loose coupling
faux-agile
flexibility
dark scrum
API
decoupling
smart endpoints and dumb burrow

microservices
tight coupling
API
decoupling
smart endpoints and dumb burrow

microservices
tight coupling
API
decoupling
smart endpoints and dumb burrow

microservices
tight coupling
API
decoupling
smart endpoints and dumb burrow

microservices
tight coupling
API
decoupling
smart endpoints and dumb burrow

microservices
tight coupling
API
decoupling
smart endpoints and dumb burrow

microservices
tight coupling
API
decoupling
smart endpoints and dumb burrow

microservices
tight coupling
API
decoupling
smart endpoints and dumb burrow

microservices
tight coupling
API
decoupling
smart endpoints and dumb burrow

microservices
tight coupling
API
decoupling
smart endpoints and dumb burrow

microservices
tight coupling
API
decoupling
smart endpoints and dumb burrow

microservices
tight coupling
API
decoupling
smart endpoints and dumb burrow