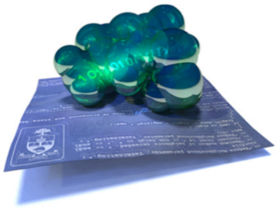
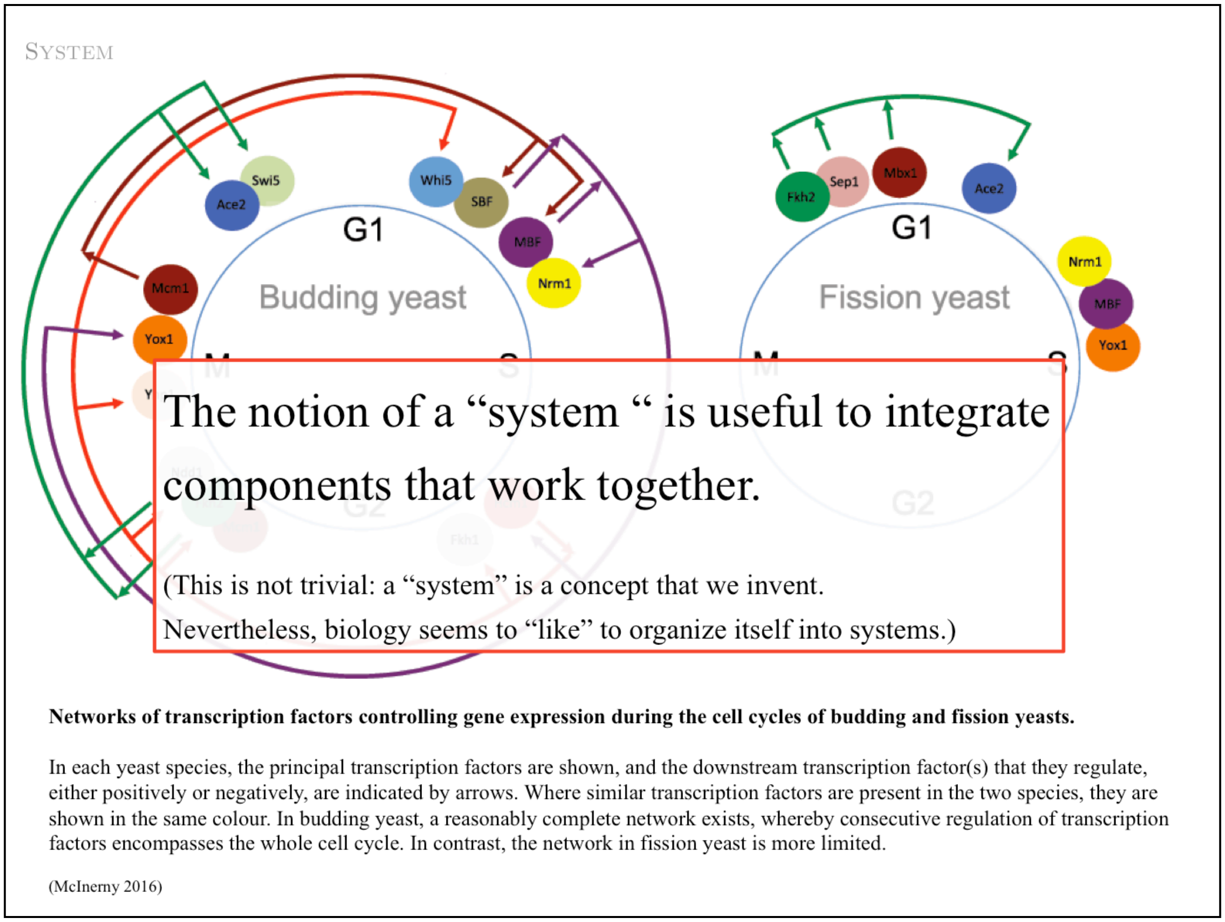
A
Bioinformatics
Course

Systems Concepts



Boris Steipe

department of biochemistry – department of molecular genetics
University of Toronto


SYSTEM

The notion of a "system " is useful to integrate components that work together.

(This is not trivial: a "system" is a concept that we invent.
Nevertheless, biology seems to "like" to organize itself into systems.)

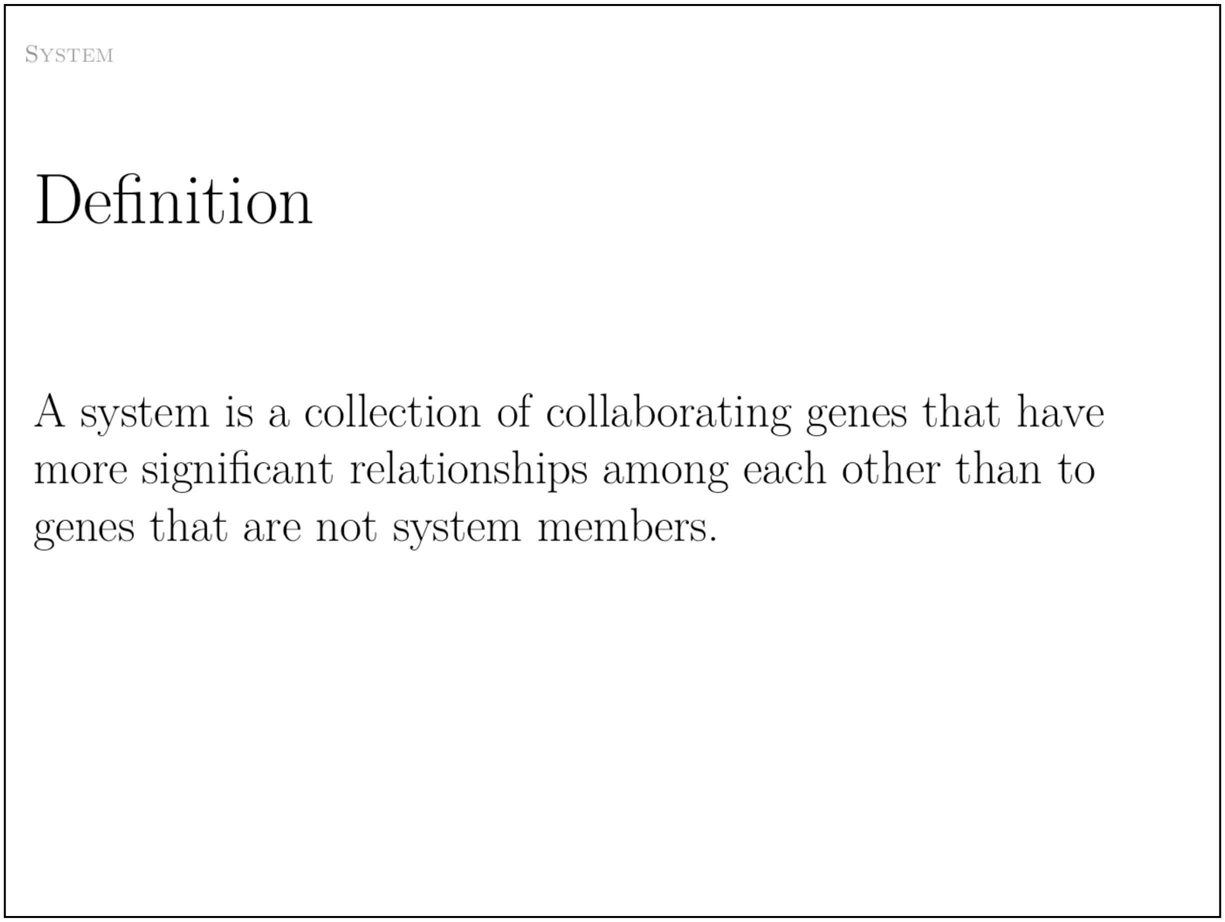
Networks of transcription factors controlling gene expression during the cell cycles of budding and fission yeasts.

In each yeast species, the principal transcription factors are shown, and the downstream transcription factor(s) that they regulate, either positively or negatively, are indicated by arrows. Where similar transcription factors are present in the two species, they are shown in the same colour. In budding yeast, a reasonably complete network exists, whereby consecutive regulation of transcription factors encompasses the whole cell cycle. In contrast, the network in fission yeast is more limited.

(McInerny 2016)

"Systems" are concepts that we use to organize our knowledge of biology.

But what is a system ... ?



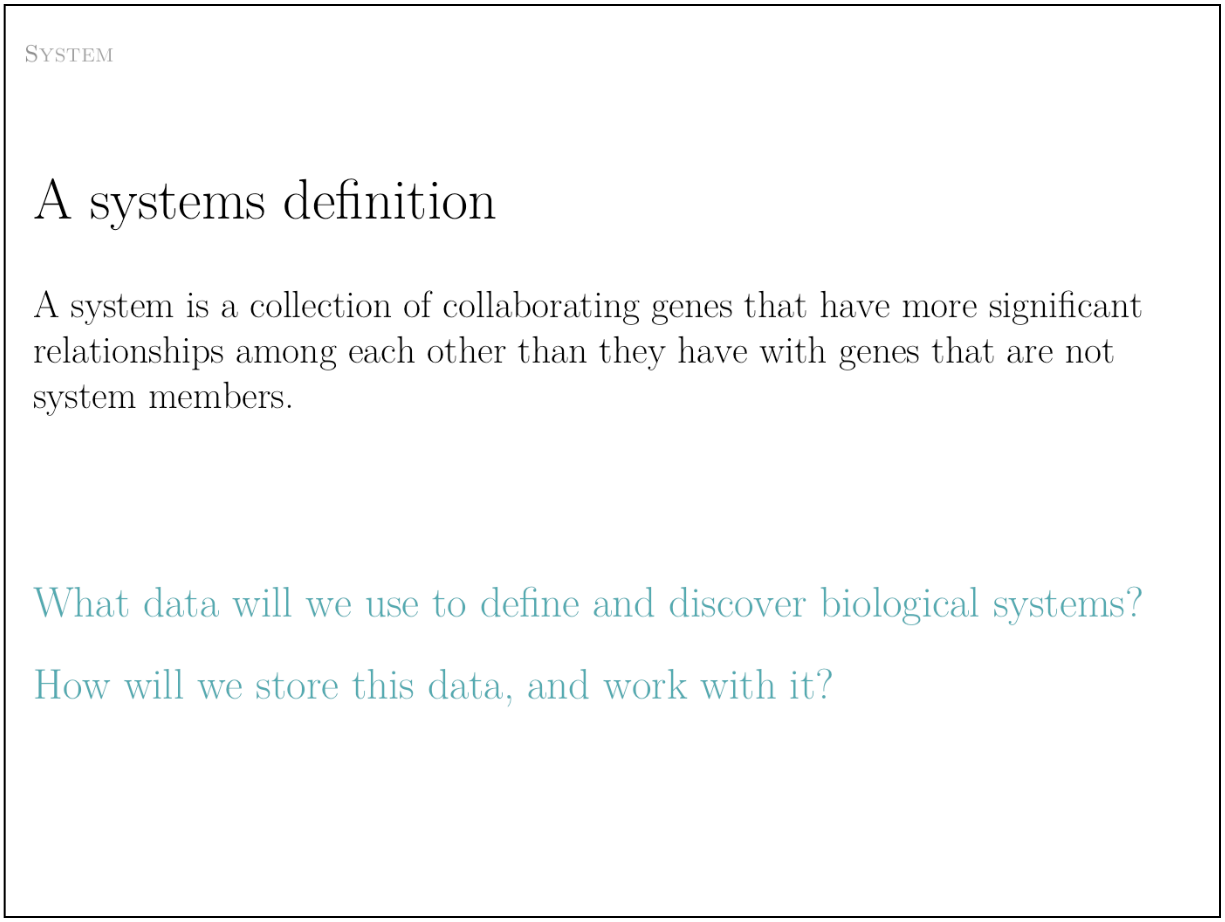
SYSTEM

Definition

A system is a collection of collaborating genes that have more significant relationships among each other than to genes that are not system members.

This is an operational definition but it has issues: what exactly do we mean by “collaborating” – collaborating towards which goal? And what do we mean by “more significant” – how do we set a threshold to decide which genes should be included and which genes should not?

The boundaries are fluid and the problem to define systems precisely is the same as defining domains in proteins, or generally clusters in data.



System

A systems definition

A system is a collection of collaborating genes that have more significant relationships among each other than they have with genes that are not system members.

What data will we use to define and discover biological systems?

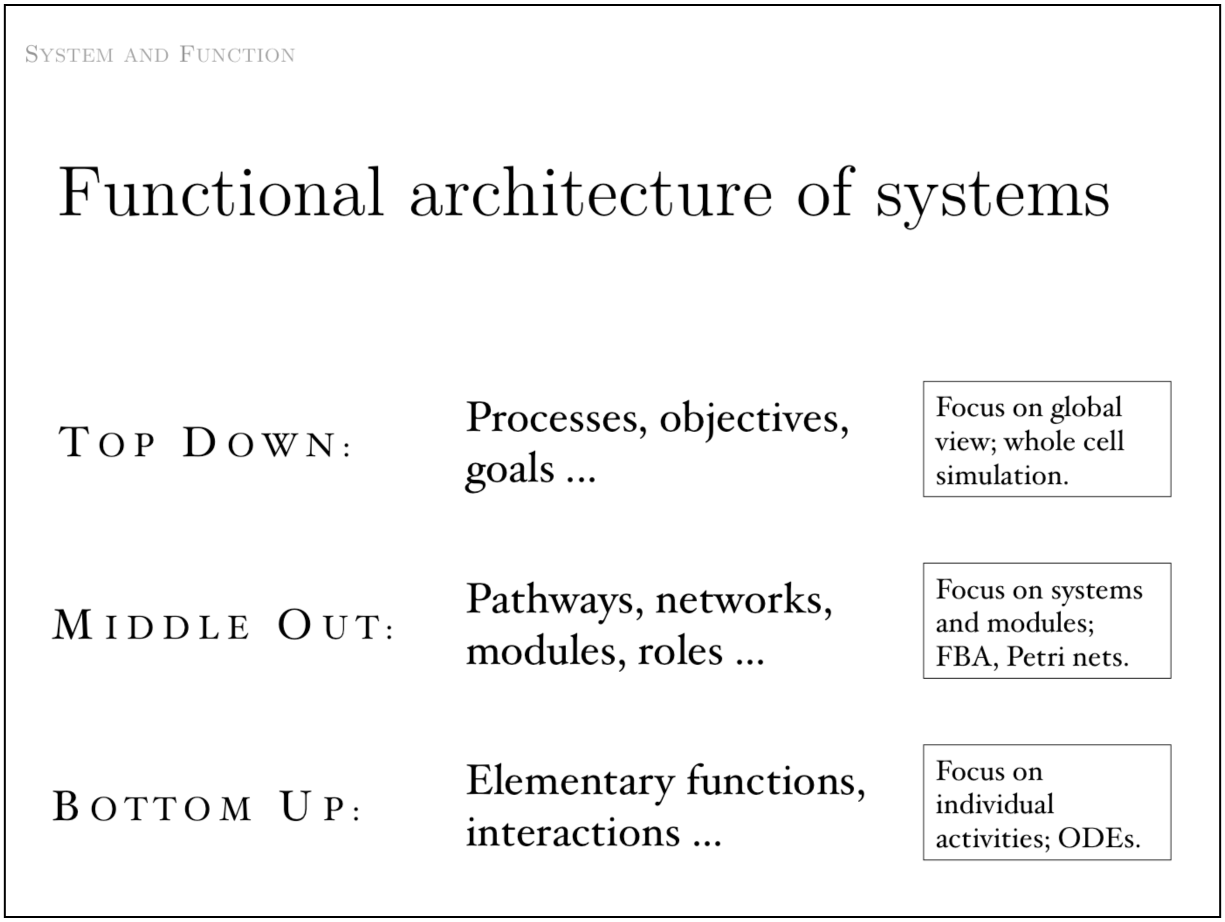
How will we store this data, and work with it?

In the sense of the definition above, a system is both a generalization of one gene's "function" and a recipe for including and excluding components.

A systems definition

A system is a collection of collaborating genes that have more significant relationships among each other than to genes that are not system members.

Our example: the yeast G1/S transition switch system.



SYSTEM AND FUNCTION

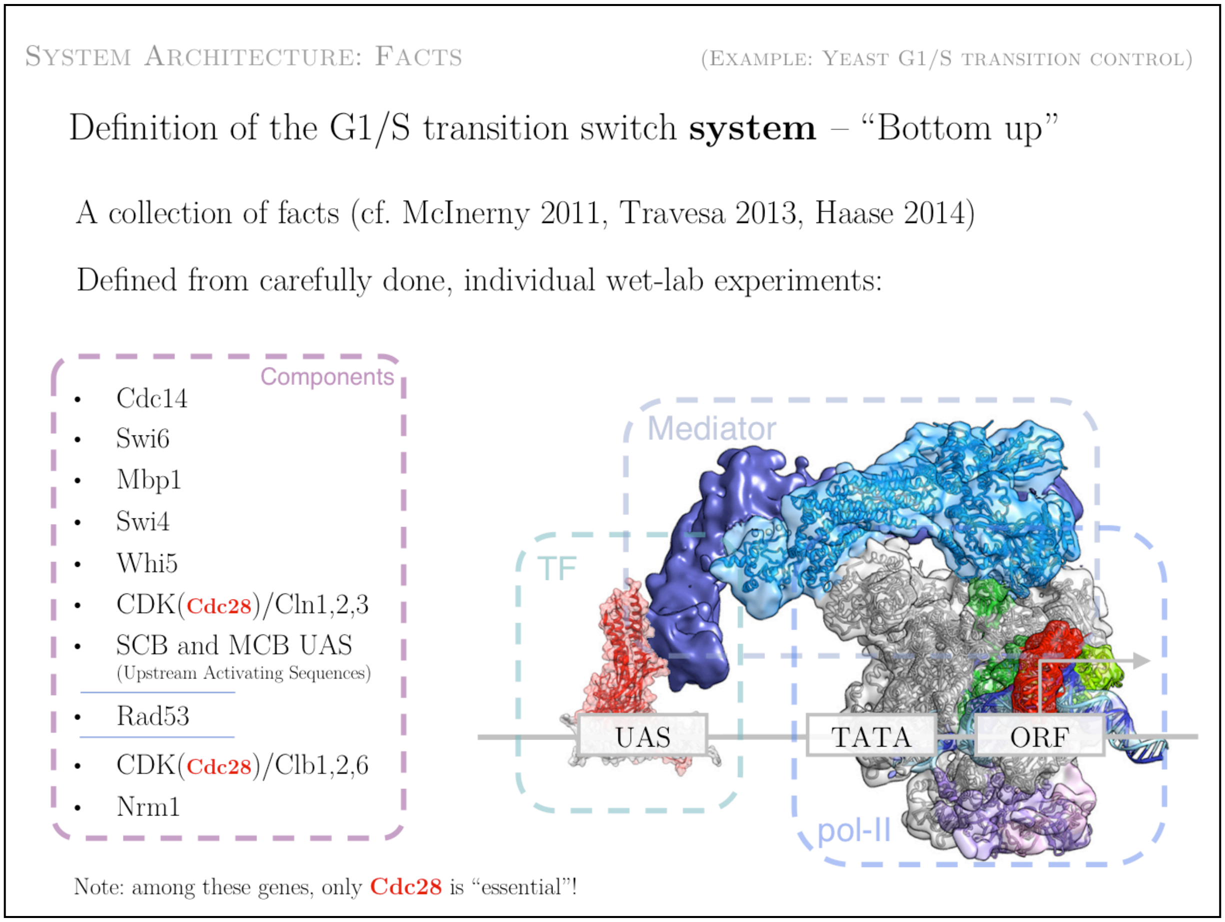
Functional architecture of systems

TOP DOWN:	Processes, objectives, goals ...	Focus on global view; whole cell simulation.
MIDDLE OUT:	Pathways, networks, modules, roles ...	Focus on systems and modules; FBA, Petri nets.
BOTTOM UP:	Elementary functions, interactions ...	Focus on individual activities; ODEs.

The goal of systems analysis is to describe the “functional architecture” of the system. This includes the components that collaborate, it also includes the roles that these components take on, and it includes the goals of the collaboration. These conceptual levels correspond to “bottom up”, “middle out” and “top down” descriptions of the system.

Our goal must be to describe systems in a way that these levels of description converge.

This is not a solved problem.



System Architecture: Facts (Example: Yeast G1/S transition control)

Definition of the G1/S transition switch **system** – “Bottom up”

A collection of facts (cf. McInerny 2011, Travesa 2013, Haase 2014)

Defined from carefully done, individual wet-lab experiments:

Components

- Cdc14
- Swi6
- Mbp1
- Swi4
- Whi5
- CDK(**Cdc28**)/Cln1,2,3
- SCB and MCB UAS (Upstream Activating Sequences)
- Rad53
- CDK(**Cdc28**)/Clb1,2,6
- Nrm1

Note: among these genes, only **Cdc28** is “essential”!

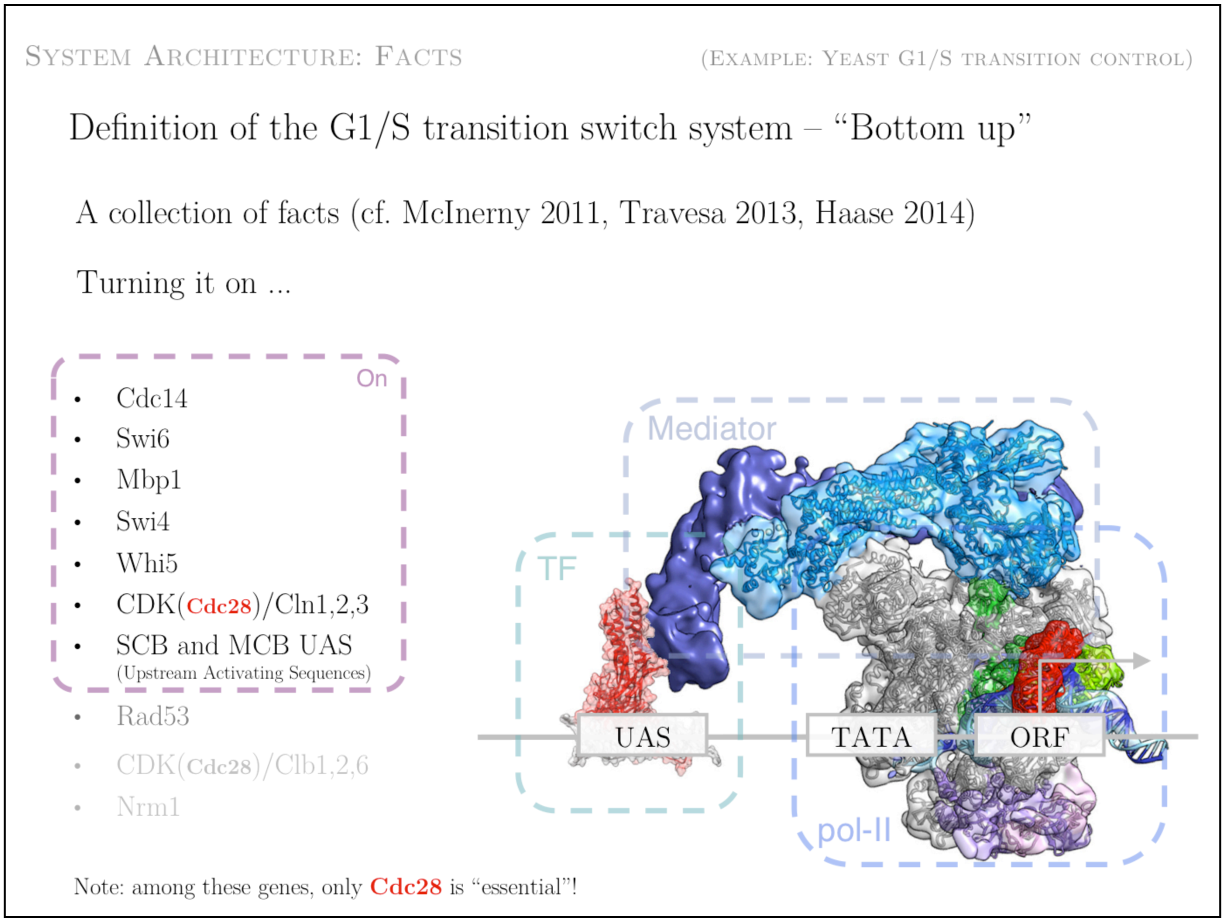
We need to start somewhere on our quest to define a G1/S transition control *system* and literature annotated genes (or genes annotated to GO terms) is as good a starting point as any. Key roles are played by the MBF and the SBF complexes that bind to a large number of UAS elements of their target genes, enabling transcription of the downstream ORF through recruitment of pol-II via the mediator complex. This is generally how regulatory transcription factors work and this mechanism provides much room for refinements in the time-course of activation – via early or later binding of transcription factors to high- resp. low-affinity variations of the UAS sequence, and refinements in the logic of activation eg. by mixing UAS to provide for mutually exclusive occupation of sites, or producing a requirement to have multiple sites occupied for effective activation.

Like many regulatory systems, we have a protein (here: Swi4 and Mbp1) that serves to integrate many different regulatory signals and pathways, which then leads to a unified signal output. We often think of these proteins as a “hub” of the system. The protein transcription factor needs to be bound and brought into an active state. Much has been written about this system but here is a handful of the most salient facts:

McInerny CJ. (2011) Cell cycle regulated gene expression in yeasts. Adv Genet. 73:51-85.

Travesa A, *et al.* (2013) Repression of G1/S transcription is mediated via interaction of the GTB motifs of Nrm1 and Whi5 with Swi6. Mol Cell Biol. 33(8):1476-86.

Haase SB and Wittenberg C.(2014) Topology and control of the cell-cycle-regulated transcriptional circuitry. Genetics. 196(1):65-90.



System Architecture: Facts (Example: Yeast G1/S transition control)

Definition of the G1/S transition switch system – “Bottom up”

A collection of facts (cf. McInerny 2011, Travesa 2013, Haase 2014)

Turning it on ...

On

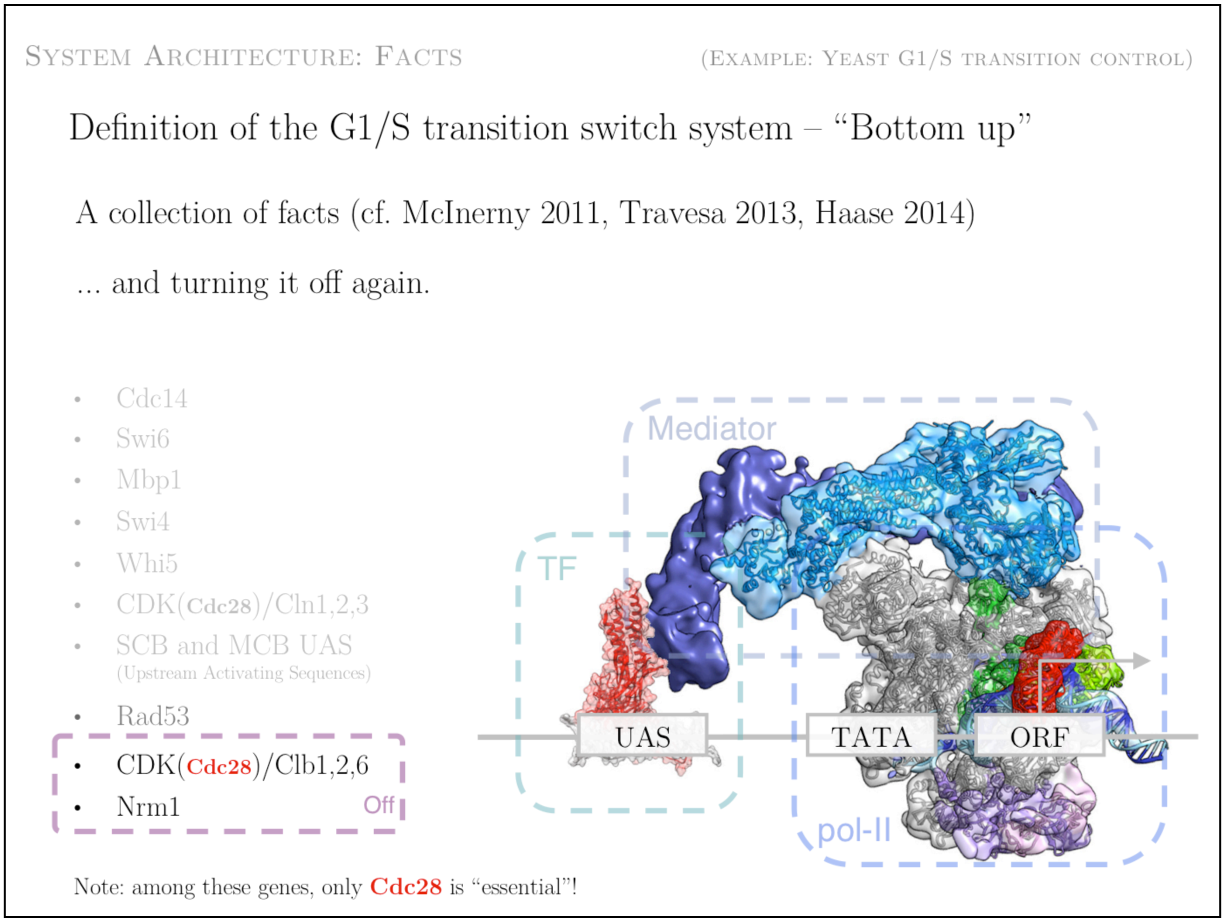
- Cdc14
- Swi6
- Mbp1
- Swi4
- Whi5
- CDK(**Cdc28**)/Cln1,2,3
- SCB and MCB UAS (Upstream Activating Sequences)
- Rad53
- CDK(**Cdc28**)/Clb1,2,6
- Nrm1

Mediator
TF
UAS
TATA
ORF
pol-II

Note: among these genes, only **Cdc28** is “essential”!

Turning it on ...

- Swi6 enters the nucleus after dephosphorylation by Cdc14.
- One wave of gene activation is associated with the MBF complex binding to MCB DNA elements. MBF consists of Mbp1 (with DNA binding activity) and Swi6 (with regulatory activity). The target genes predominantly function in DNA replication.
- Another wave of gene activation is associated with the SBF complex binding to SCB DNA elements. SBF consists of Swi4 (with DNA binding activity) and Swi6 (with regulatory activity). The target genes predominantly function in the preparation of budding, cell-wall biosynthesis etc.
- Swi4 expression itself is regulated through ECB motifs in its promoter: this causes expression during G1 and provides a basal amountof Swi4 transcription factor to prime its activity. Once Swi4 is switched on, it targets its own gene’s promoter, thus providing a positive feedback loop.
- Binding of MBF/SBF to their respective UAS does in itself not lead to activation. Activation requires the action of a CDK (cyclin dependent kinase) complex of Cdc28 and Cln3.
- The activation of MBF is obscure, but depends on a Cdc28/Cln CDK.
- SBF is activated when phosphorylation by CDK removes the inhibitor Whi5 which is normally bound to Swi6 in the SBF complex. Phosphorylated Whi5 is excluded from the nucleus.
- The actual process of replication is driven by the MBF/SBF target genes Clb5/Clb6. These are initially kept inactive through binding of Sic1, but Sic1 is destroyed after phosphorylation by CDK1 (under MBF control), later in S phase.



System Architecture: Facts (Example: Yeast G1/S transition control)

Definition of the G1/S transition switch system – “Bottom up”

A collection of facts (cf. McInerny 2011, Travesa 2013, Haase 2014)

... and turning it off again.

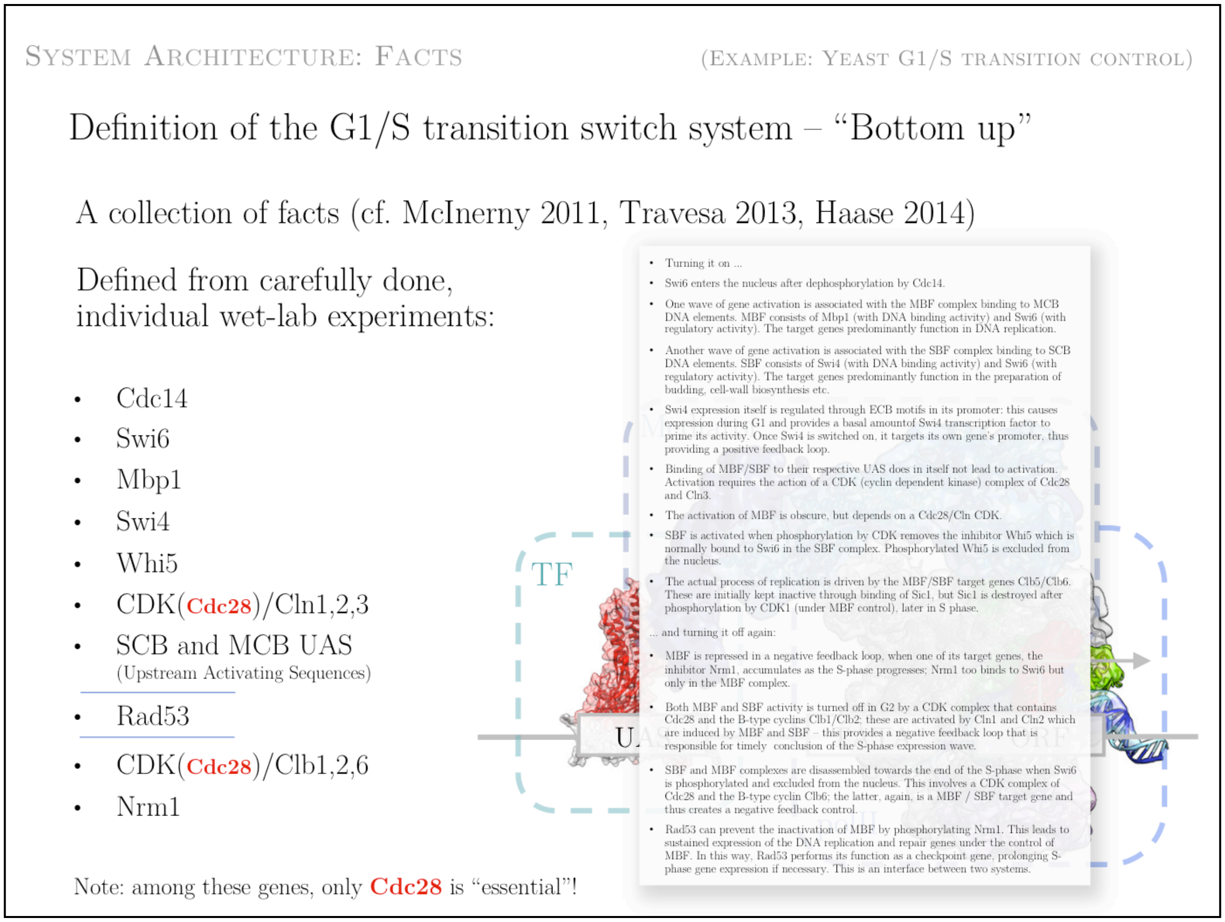
- Cdc14
- Swi6
- Mbp1
- Swi4
- Whi5
- CDK(**Cdc28**)/Cln1,2,3
- SCB and MCB UAS (Upstream Activating Sequences)
- Rad53
- CDK(**Cdc28**)/Clb1,2,6
- Nrm1

Off

Note: among these genes, only **Cdc28** is “essential”!

... and turning it off again:

- MBF is repressed in a negative feedback loop, when one of its target genes, the inhibitor Nrm1, accumulates as the S-phase progresses; Nrm1 too binds to Swi6 but only in the MBF complex.
- Both MBF and SBF activity is turned off in G2 by a CDK complex that contains Cdc28 and the B-type cyclins Clb1/Clb2; these are activated by Cln1 and Cln2 which are induced by MBF and SBF – this provides a negative feedback loop that is responsible for timely conclusion of the S-phase expression wave.
- SBF and MBF complexes are disassembled towards the end of the S-phase when Swi6 is phosphorylated and excluded from the nucleus. This involves a CDK complex of Cdc28 and the B-type cyclin Clb6; the latter, again, is a MBF / SBF target gene and thus creates a negative feedback control.
- Rad53 can prevent the inactivation of MBF by phosphorylating Nrm1. This leads to sustained expression of the DNA replication and repair genes under the control of MBF. In this way, Rad53 performs its function as a checkpoint gene, prolonging S-phase gene expression if necessary. This is an interface between two systems.



SYSTEM ARCHITECTURE: FACTS (EXAMPLE: YEAST G1/S TRANSITION CONTROL)

Definition of the G1/S transition switch system – "Bottom up"

A collection of facts (cf. McInerny 2011, Travesa 2013, Haase 2014)

Defined from carefully done, individual wet-lab experiments:

- Cdc14
- Swi6
- Mbp1
- Swi4
- Whi5
- CDK(**Cdc28**)/Cln1,2,3
- SCB and MCB UAS (Upstream Activating Sequences)
- Rad53
- CDK(**Cdc28**)/Clb1,2,6
- Nrm1

Note: among these genes, only **Cdc28** is "essential"!

- Turning it on ...
- Swi6 enters the nucleus after dephosphorylation by Cdc14.
- One wave of gene activation is associated with the MBF complex binding to MCB DNA elements. MBF consists of Mbp1 (with DNA binding activity) and Swi6 (with regulatory activity). The target genes predominantly function in DNA replication.
- Another wave of gene activation is associated with the SBF complex binding to SCB DNA elements. SBF consists of Swi4 (with DNA binding activity) and Swi6 (with regulatory activity). The target genes predominantly function in the preparation of budding, cell-wall biosynthesis etc.
- Swi4 expression itself is regulated through ECB motifs in its promoter: this causes expression during G1 and provides a basal amountof Swi4 transcription factor to prime its activity. Once Swi4 is switched on, it targets its own gene's promoter, thus providing a positive feedback loop.
- Binding of MBF/SBF to their respective UAS does in itself not lead to activation. Activation requires the action of a CDK (cyclin dependent kinase) complex of Cdc28 and Cln3.
- The activation of MBF is obscure, but depends on a Cdc28/Cln CDK.
- SBF is activated when phosphorylation by CDK removes the inhibitor Whi5 which is normally bound to Swi6 in the SBF complex. Phosphorylated Whi5 is excluded from the nucleus.
- The actual process of replication is driven by the MBF/SBF target genes Clb5/Clb6. These are initially kept inactive through binding of Sic1, but Sic1 is destroyed after phosphorylation by CDK1 (under MBF control), later in S phase.

... and turning it off again:

- MBF is repressed in a negative feedback loop, when one of its target genes, the inhibitor Nrm1, accumulates as the S-phase progresses; Nrm1 too binds to Swi6 but only in the MBF complex.
- Both MBF and SBF activity is turned off in G2 by a CDK complex that contains Cdc28 and the B-type cyclins Clb1/Clb2; these are activated by Cln1 and Cln2 which are induced by MBF and SBF – this provides a negative feedback loop that is responsible for timely conclusion of the S-phase expression wave.
- SBF and MBF complexes are disassembled towards the end of the S-phase when Swi6 is phosphorylated and excluded from the nucleus. This involves a CDK complex of Cdc28 and the B-type cyclin Clb6; the latter, again, is a MBF / SBF target gene and thus creates a negative feedback control.
- Rad53 can prevent the inactivation of MBF by phosphorylating Nrm1. This leads to sustained expression of the DNA replication and repair genes under the control of MBF. In this way, Rad53 performs its function as a checkpoint gene, prolonging S-phase gene expression if necessary. This is an interface between two systems.

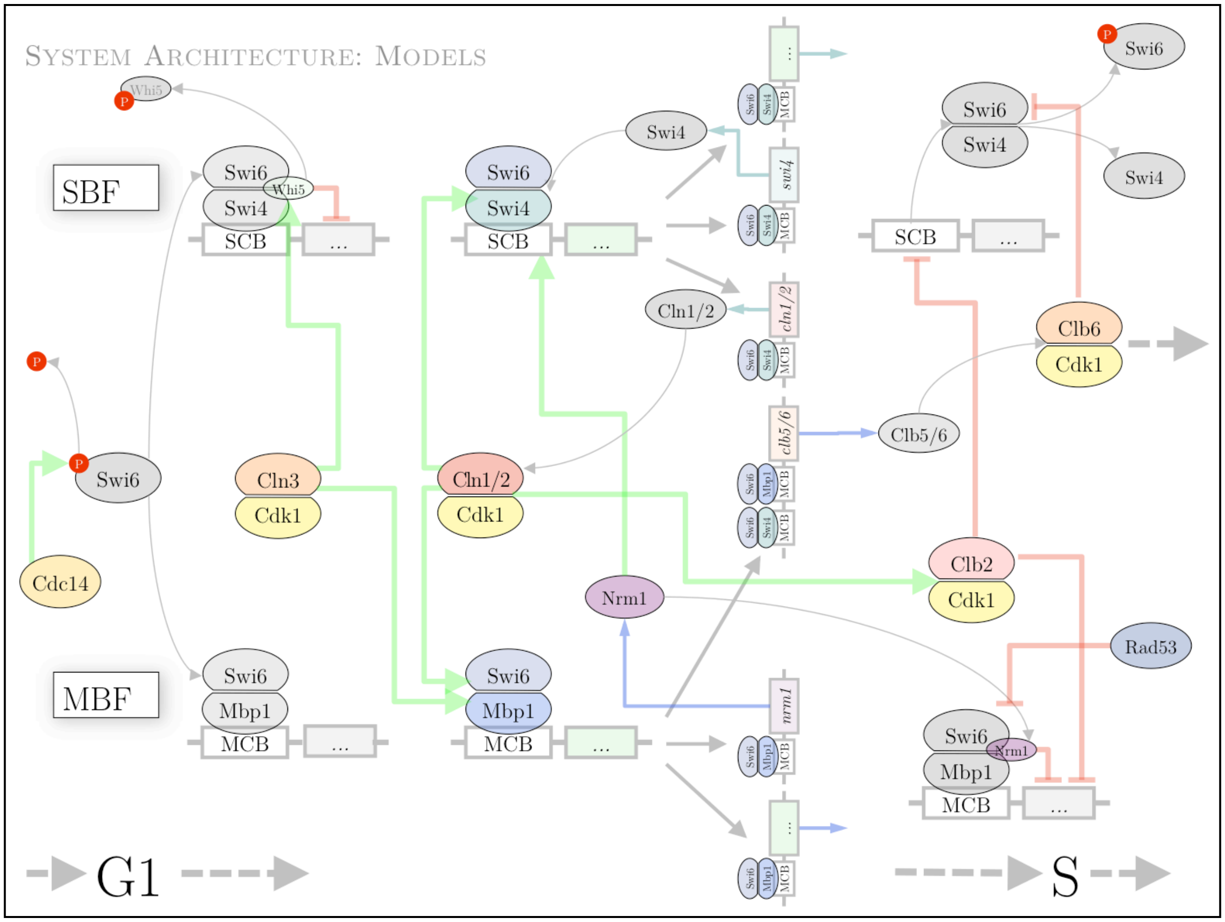
Though factually accurate as far as I can tell, there are two things wrong with this description that make it only marginally useful:

(i) *Understanding* this collection of facts about symbols is not intuitive. It is remarkably difficult to organize this small body of knowledge. It is also remarkably difficult to commit it to memory.

(ii) Key concepts and ideas are *missing* from the description: where is the notion of a commited START (or restriction point); where do we mention that this network of interactions constitutes a "switch"?

We have to ask: is there a better way to represent this knowledge? In particular, a way to focus on the conceptual aspects of the system, since these will help us construct a mental map in which we can integrate detailed information from our collection of facts.

Let us first see how existing databases address this.



I have drawn out the facts in a style that is popular among authors in molecular biology and the sketch will evoke others you have seen on the topic in the reviews I have mentioned above, in particular Haase (2014).

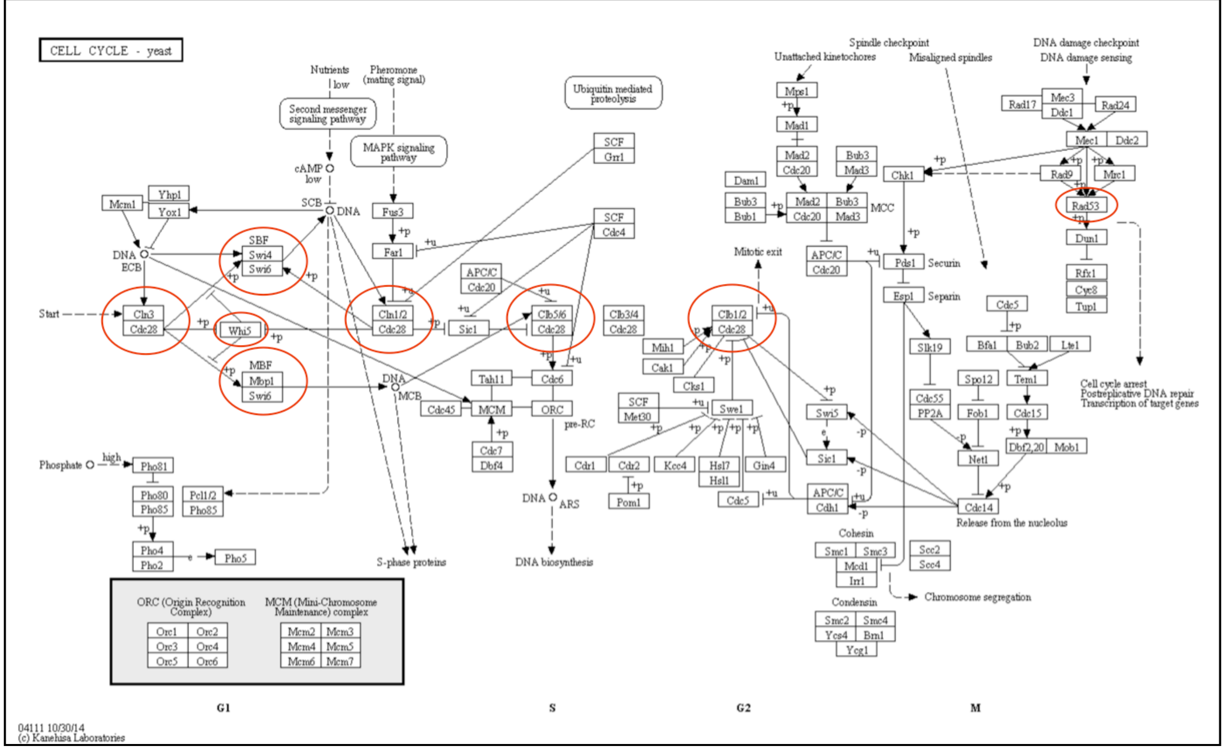
The sketch summarizes the facts by drawing out the activating and inhibiting relationships between the transcription units and their products with green and red lines respectively, and protein movement with grey curves. Active complexes are colored and inactive proteins are grey. In general, changes progress from G1 to S, left to right.

Although the relationships are now visualized, I would consider this sketch to be just as impenetrable as the description of the facts, albeit for different reasons. Indeed, a defining element of the *structure* of the cell-cycle are specific points such as START in yeast (or the "restriction point" in higher eukaryotes) that represent an irreversible commitment to replication. Such commitment is not "a gene". Nor is it obvious which of the connections comprise feed-forward loops that switch the system into the "on" state, and feedback loops that switch it "off" again. Finally it is very hard to figure out which of the components provide "internal" control, and which ones constitute the input and output of the system. These points are the most severe criticism of sketches like these (and enumeration of components + activities): **the concept behind the system, the purpose, or meaning if you will is not expressed explicitly**.

We need a representation of the system *architecture*. In the language of sytems engineers, a list of proteins / complexes would be a "decompositional view", a view that captures the collaborating components but says little about their roles, and virtually nothing about the concept or purpose that is realized through these components. Let us compare some alternative diagramming conventions.

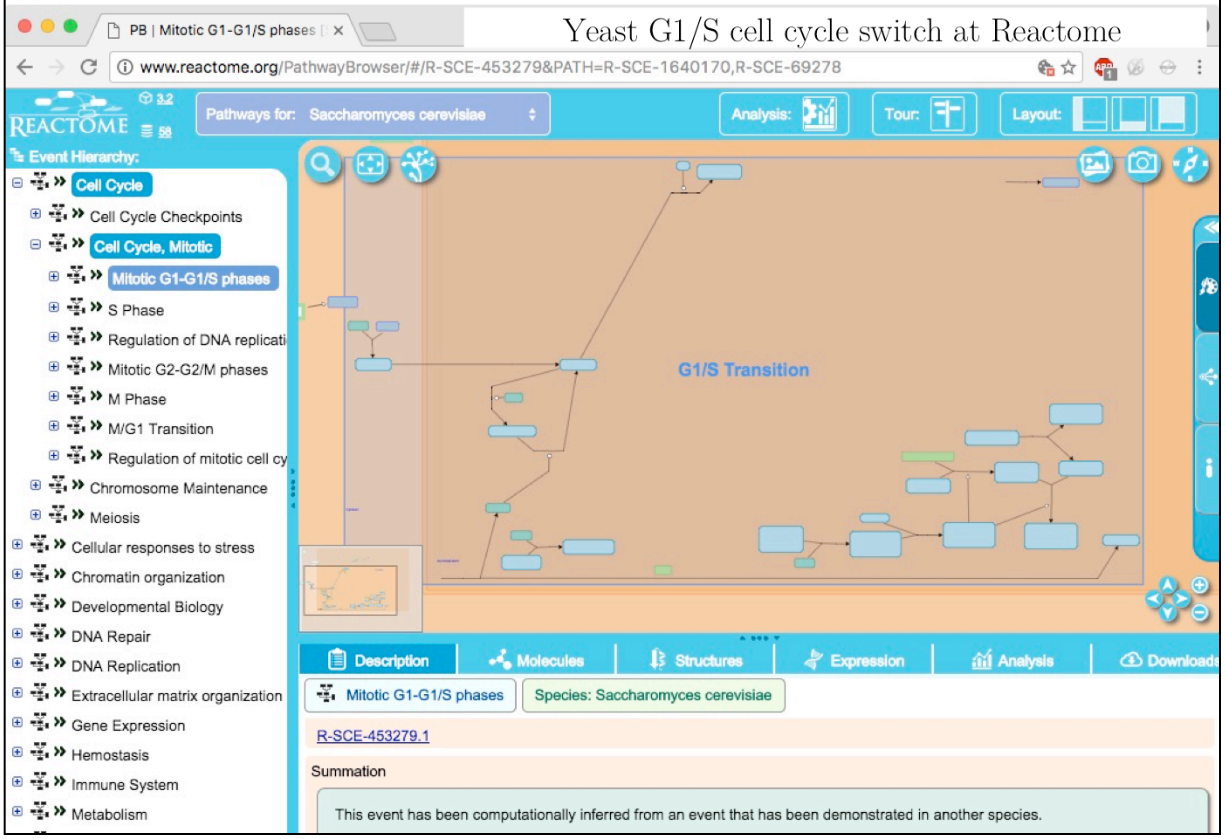
System Architecture: Models (Example: Yeast G1/S transition control)

Yeast G1/S cycle switch in KEGG



Many of the facts from our list are not represented in the KEGG pathway. The activation of Swi6 by the Cdc14 phosphatase, and its inactivation by Clb2 is missing. Nrm1 is entirely absent. The roles of Clb1/2 and Rad53 in modulating the switch are only vaguely described. The crucial feedback and feedforward loops are not readily apparent.

System Architecture: Models (Example: Yeast G1/S transition control)



Yeast G1/S cell cycle switch at Reactome

www.reactome.org/PathwayBrowser/#/R-SCE-453279&PATH=R-SCE-1640170,R-SCE-69278

Mitotic G1-G1/S phases

Species: Saccharomyces cerevisiae

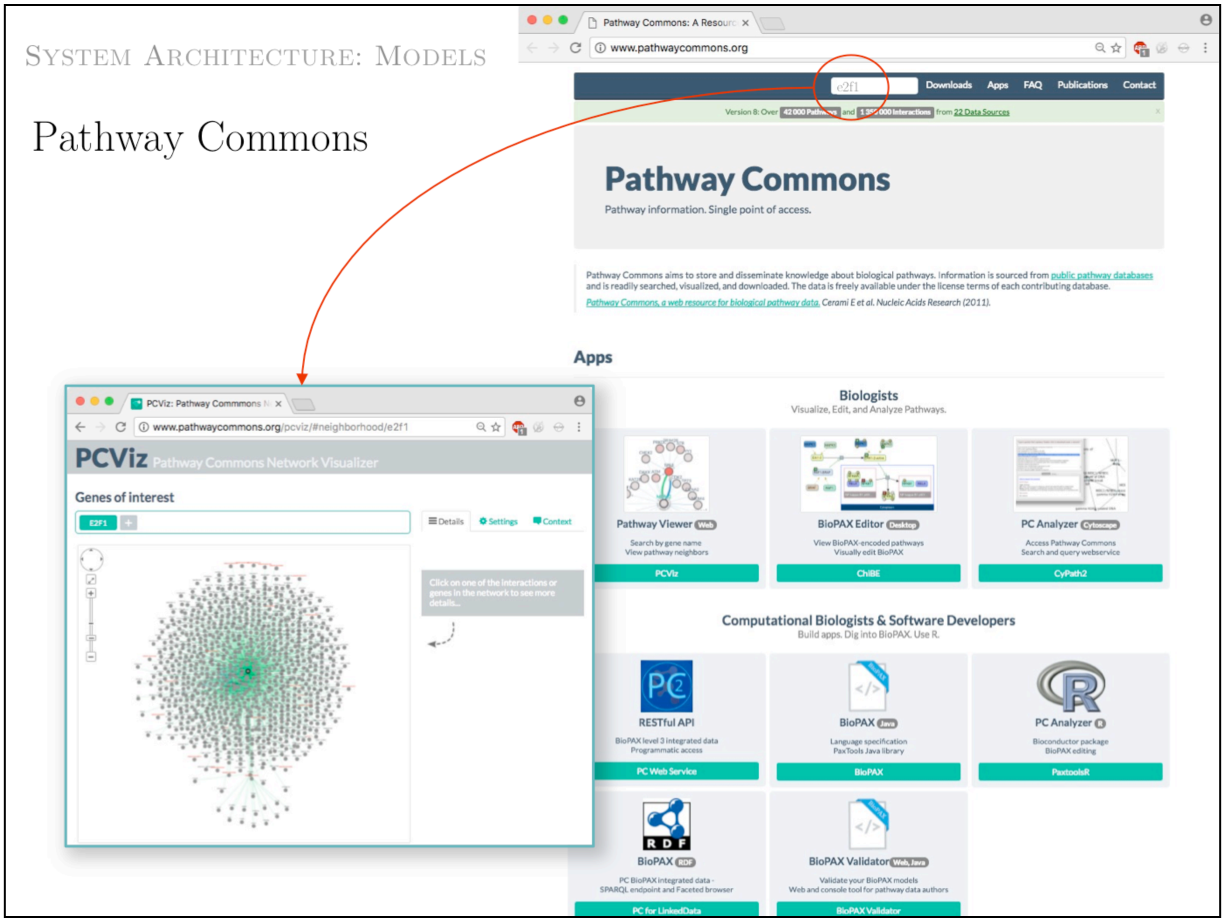
R-SCE-453279.1

Summation

This event has been computationally inferred from an event that has been demonstrated in another species.

The Reactome pathway database is an international collaboration that includes the EBI, CSH laboratories and OICR. The interface is carefully engineered – but I was not able to map our yeast cell cycle facts to the schematic; the nodes displayed here do not reflect primary data but are inferred by computational means from human annotations. Merely judging from the topology however, it is apparent that the schematic shares other databases' issues with bottom-up approaches.

http://www.reactome.org/



System Architecture: Models

Pathway Commons

Pathway Commons: A Resourc

www.pathwaycommons.org

e2f1

Version 8: Over 42 000 Pathways and 1 35 000 Interactions from 22 Data Sources

Pathway Commons

Pathway information. Single point of access.

Pathway Commons aims to store and disseminate knowledge about biological pathways. Information is sourced from public pathway databases and is readily searched, visualized, and downloaded. The data is freely available under the license terms of each contributing database.

Pathway Commons, a web resource for biological pathway data. *Cerami E et al. Nucleic Acids Research (2011).*

Apps

Biologists

Visualize, Edit, and Analyze Pathways.

Pathway Viewer (Web): Search by gene name; View pathway neighbors

BioPAX Editor (Desktop): View BioPAX-encoded pathways; Visually edit BioPAX

PC Analyzer (Cytoscape): Access Pathway Commons; Search and query webservice

Computational Biologists & Software Developers

Build apps. Dig into BioPAX. Use R.

RESTful API: BioPAX level 3 integrated data; Programmatic access

BioPAX (Java): Language specification; PaxTools Java library

PC Analyzer (R): Bioconductor package; BioPAX editing

BioPAX (RDF): PC BioPAX integrated data - SPARQL endpoint and Faceted browser

BioPAX Validator (Web, Java): Validate your BioPAX models; Web and console tool for pathway data authors

PCViz: Pathway Commmons N

www.pathwaycommons.org/pcviz/#neighborhood/e2f1

PCViz Pathway Commons Network Visualizer

Genes of interest

E2F1

Click on one of the interactions or genes in the network to see more details...

Pathway commons is an integrated access point to biological pathway information from multiple sources. The researchers behind this endeavour have been especially active in developing BioPAX – a pathway information exchange language – that allows automated import and cross-referencing of pathway data. This type of infrastructure is indispensible for making distributed efforts at data collecion, annotation and curation interoperable.

The example above shows the network of interactors that is retrieved for a query for the human analogue of Swi4 – E2f1. (A direct search for yeast genes did not retrieve results.)

http://www.pathwaycommons.org/

http://biopax.org

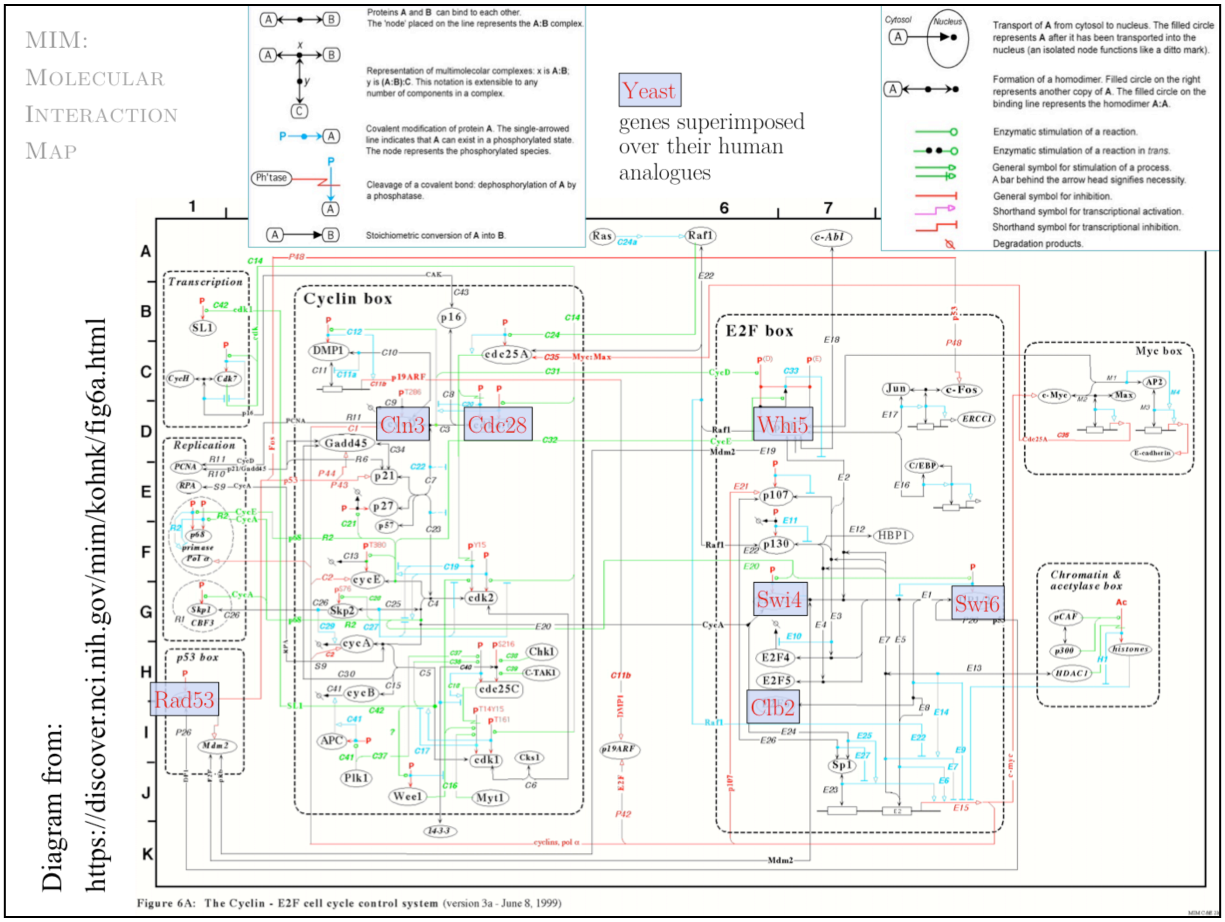

MIM:
Molecular
Interaction
Map
Yeast
genes superimposed over their human analogues
Proteins A and B can bind to each other. The 'node' placed on the line represents the A:B complex.
Representation of multimolecular complexes: x is A:B; y is (A:B):C. This notation is extensible to any number of components in a complex.
Covalent modification of protein A. The single-arrowed line indicates that A can exist in a phosphorylated state. The node represents the phosphorylated species.
Cleavage of a covalent bond: dephosphorylation of A by a phosphatase.
Stoichiometric conversion of A into B.
Transport of A from cytosol to nucleus. The filled circle represents A after it has been transported into the nucleus (an isolated node functions like a ditto mark).
Formation of a homodimer. Filled circle on the right represents another copy of A. The filled circle on the binding line represents the homodimer A:A.
Enzymatic stimulation of a reaction.
Enzymatic stimulation of a reaction in trans.
General symbol for stimulation of a process.
A bar behind the arrow head signifies necessity.
General symbol for inhibition.
Shorthand symbol for transcriptional activation.
Shorthand symbol for transcriptional inhibition.
Degradation products.
Transcription
Cyclin box
E2F box
Myc box
Replication
p53 box
Chromatin & acetylase box
Cln3
Cdc28
Whi5
Swi4
Swi6
Clb2
Rad53
Diagram from:
https://discover.nci.nih.gov/mim/kohnk/fig6a.html

Figure 6A: The Cyclin - E2F cell cycle control system (version 3a - June 8, 1999)

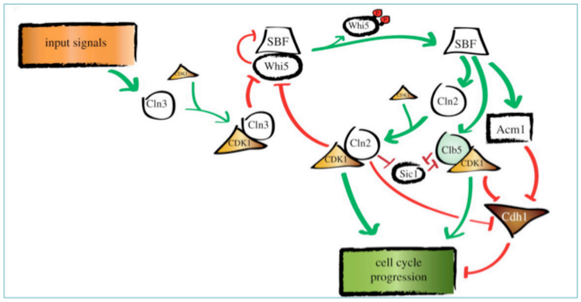
This style of molecular map was devised by Kurt Kohn of the US National Cancer Institute. It aims to unambiguously represent networks of interactions, PTMs and their enzyme/substrate relationships. Each molecular species is represented only once. I have highlighted those points in the network where human proteins have the same role as the yeast proteins of the G1/S switch system. There is a clear effort to organize components into modules like “Cyclin box”, “E2F box”, but this structural definition of modules does not map well to functional roles. It’s as if you were to lump the engine and the windshield washer of a car together as “parts under the hood”, and the transmission, differential and brakes into “parts attached to the wheel”. Once again the crucial feedforward and feedback loops are not easily identified and thus the conceptual dimension of the systems is not easily inferred from this representation.

Kohn KW, Aladjem MI, Weinstein JN, Pommier Y. (2006) Molecular interaction maps of bioregulatory networks: a general rubric for systems biology. Mol Biol Cell. 17(1):1-13.

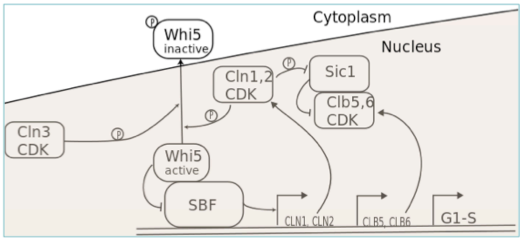
System Architecture: Models

(Example: Yeast G1/S transition control)

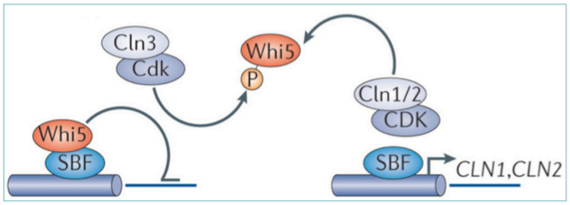
Other Depictions:



Cross *et al.* (2011)



Wikipedia: Cln3



Bertoli *et al.* (2013)

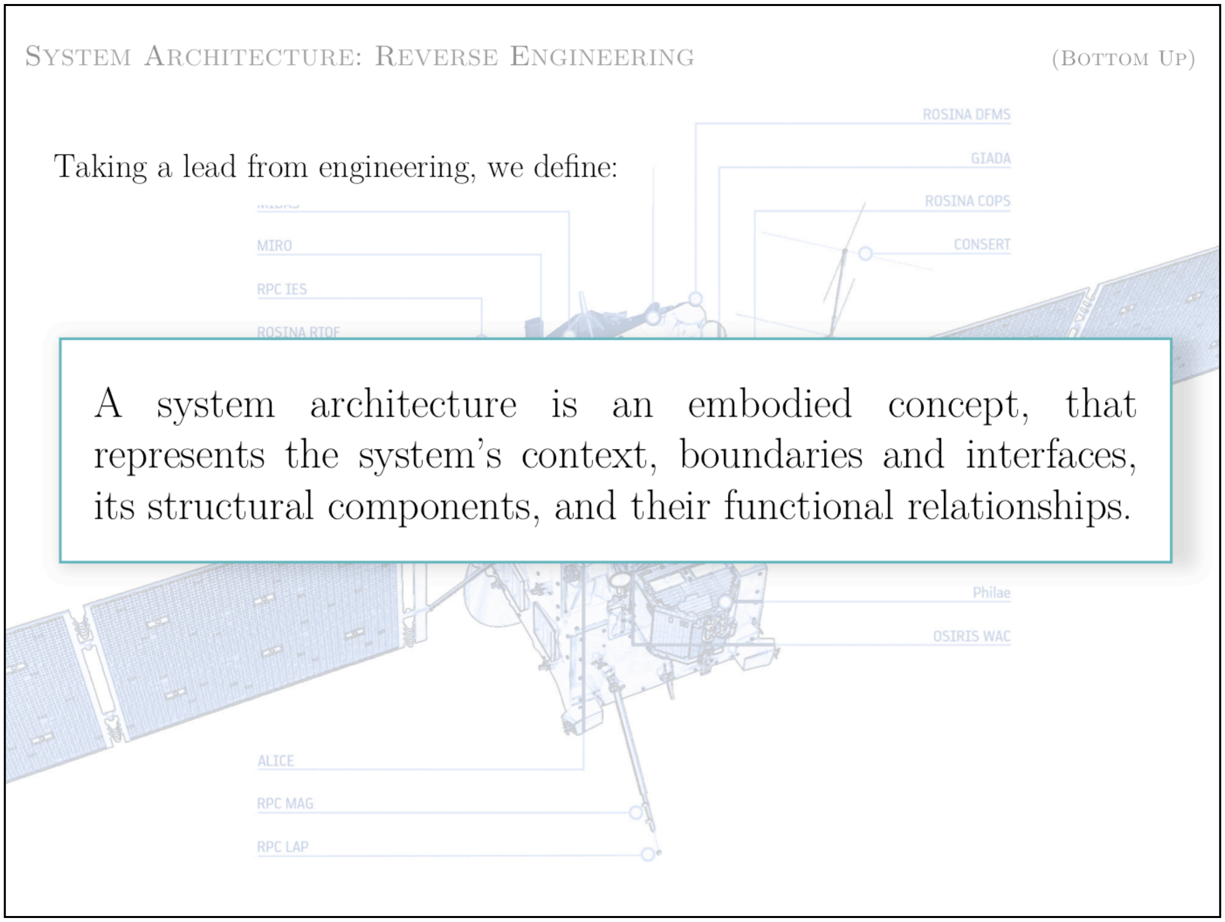
Some other examples via a Google image search.

None of the examples encountered so far are satisfactory in capturing the essential *architecture* of the system.

Cross FR, Buchler NE, and Skotheim JM (2011) Evolution of networks and sequences in eukaryotic cell cycle control. Philos Trans R Soc Lond B Biol Sci. 366(1584):3532-3544.

Bertoli C, Skotheim JM, and & de Bruin RAM (2013) Control of cell cycle transcription during G1 and S phases. Nature Reviews Molecular Cell Biology 14:518–528.

https://en.wikipedia.org/wiki/Cln3



SYSTEM ARCHITECTURE: REVERSE ENGINEERING (BOTTOM UP)

Taking a lead from engineering, we define:

A system architecture is an embodied concept, that represents the system's context, boundaries and interfaces, its structural components, and their functional relationships.

System architectures are often employed in engineering. Let us consider how the definition of a system architecture can be obtained from "reverse engineering" the biological facts.

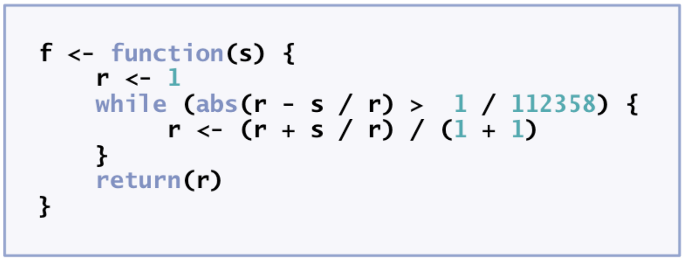
Reverese engineering means arriving at a high-level description of a system from considering its elementary components.

The "system architecture" definition is a guide to reverse engineering since it defines the aspects of the system we need to define and map to its observables.

System Architecture: Reverse Engineering (Bottom Up)

A system architecture is an embodied concept, that represents the system's context, boundaries and interfaces, its structural components, and their functional relationships.

Consider this "system":



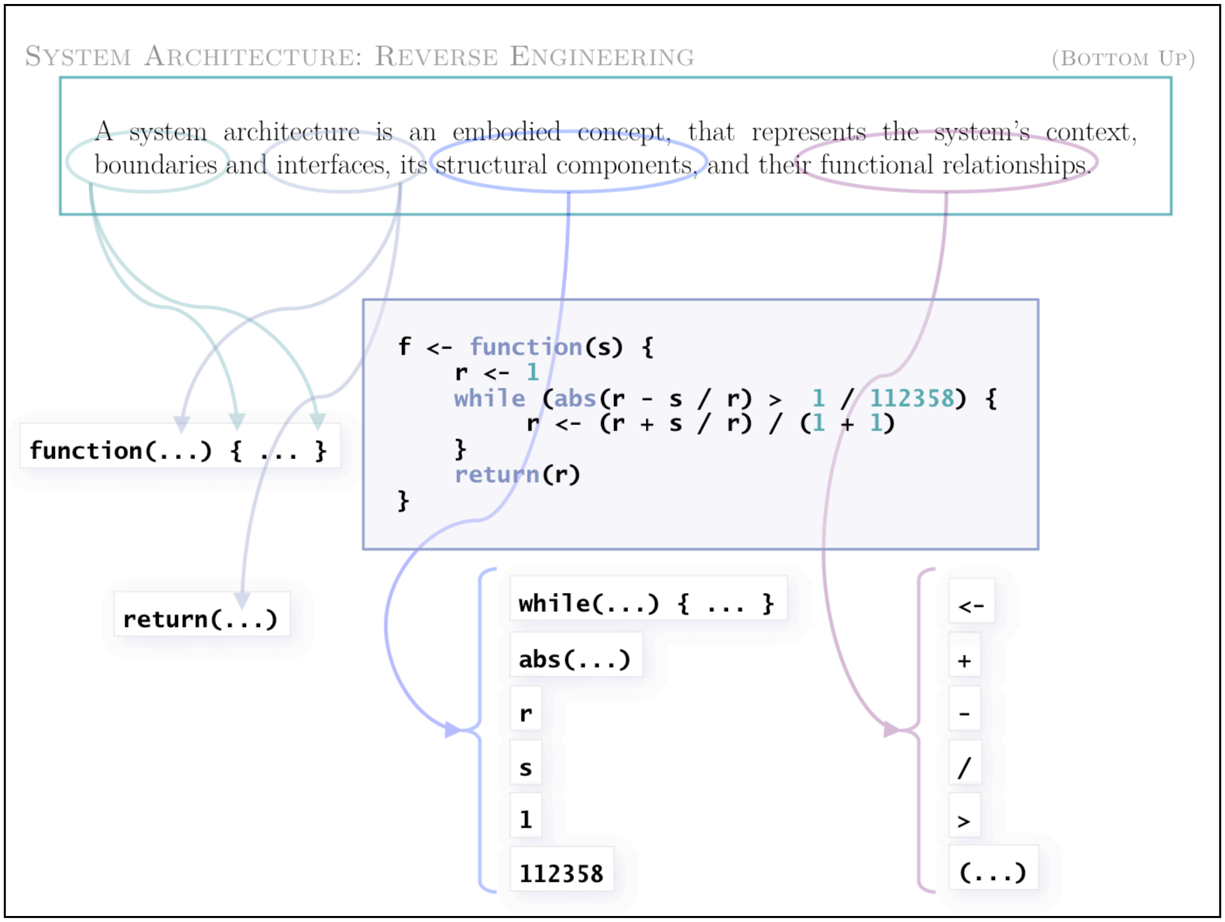
```
f <- function(s) {
    r <- 1
    while (abs(r - s / r) >  1 / 112358) {
         r <- (r + s / r) / (1 + 1)
    }
    return(r)
}
```

What does it do?

Forward engineering is easier than reverse engineering. Data does not explain itself.

As an example of how we apply our "system architecture" to reverse engineering, we consider an algorithm expressed as a computer function written in R.

Spend some time trying to understand this code. Then express the purpose of this function with one sentence.



System Architecture: Reverse Engineering (Bottom Up)

A system architecture is an embodied concept, that represents the system's context, boundaries and interfaces, its structural components, and their functional relationships.

```
f <- function(s) {
    r <- 1
    while (abs(r - s / r) >  1 / 112358) {
        r <- (r + s / r) / (1 + 1)
    }
    return(r)
}
```

`function(...) { ... }`

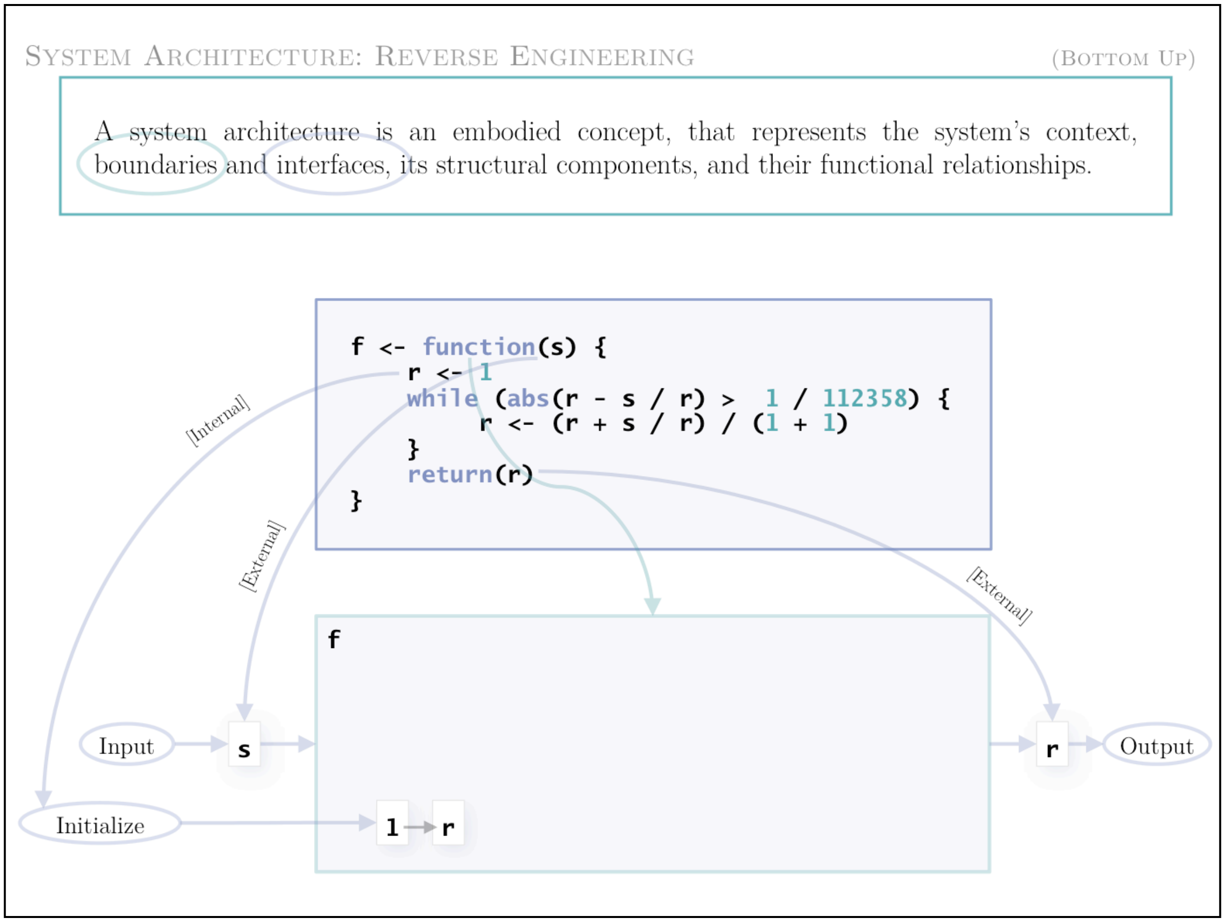
`return(...)`

`while(...) { ... }`
`abs(...)`
`r`
`s`
`1`
`112358`

`<-`
`+`
`-`
`/`
`>`
`(...)`

Understand that an enumeration of components (the "genome" and "proteome") is a crucial beginning, but does not tell us the purpose (the underlying *concept*) of the system, nor what roles the components have.

Here we decompose and enumerate the components, and categorize them as boundaries, interfaces, structural components and functional relationships.

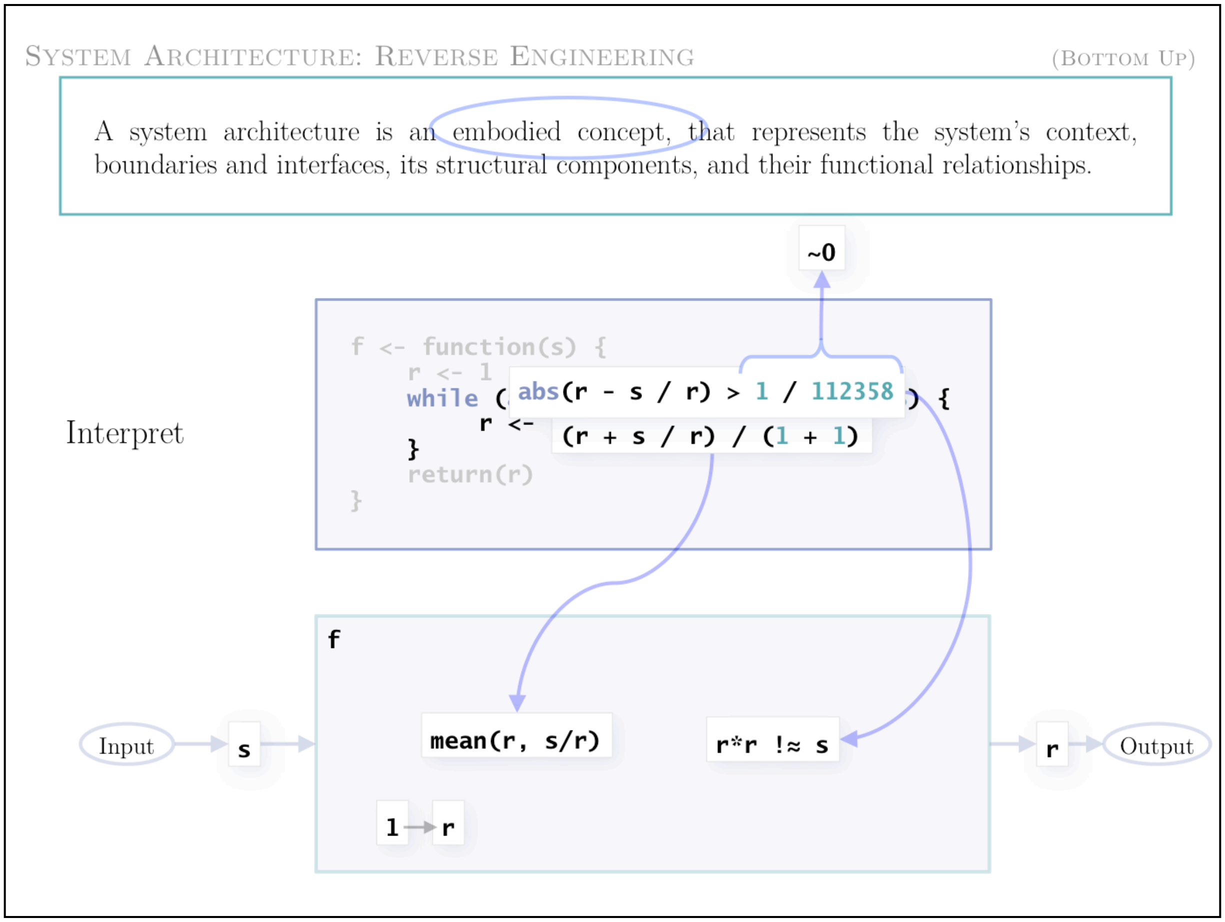


System Architecture: Reverse Engineering (Bottom Up)

A system architecture is an embodied concept, that represents the system's context, boundaries and interfaces, its structural components, and their functional relationships.

```
f <- function(s) {
    r <- 1
    while (abs(r - s / r) >  1 / 112358) {
         r <- (r + s / r) / (1 + 1)
    }
    return(r)
}
```

A system can have internal and external interfaces. We may abstract the external interfaces as *input* and *output*. Internal interfaces may *create / constitute* components and *dispose / destroy* them.



System Architecture: Reverse Engineering (Bottom Up)

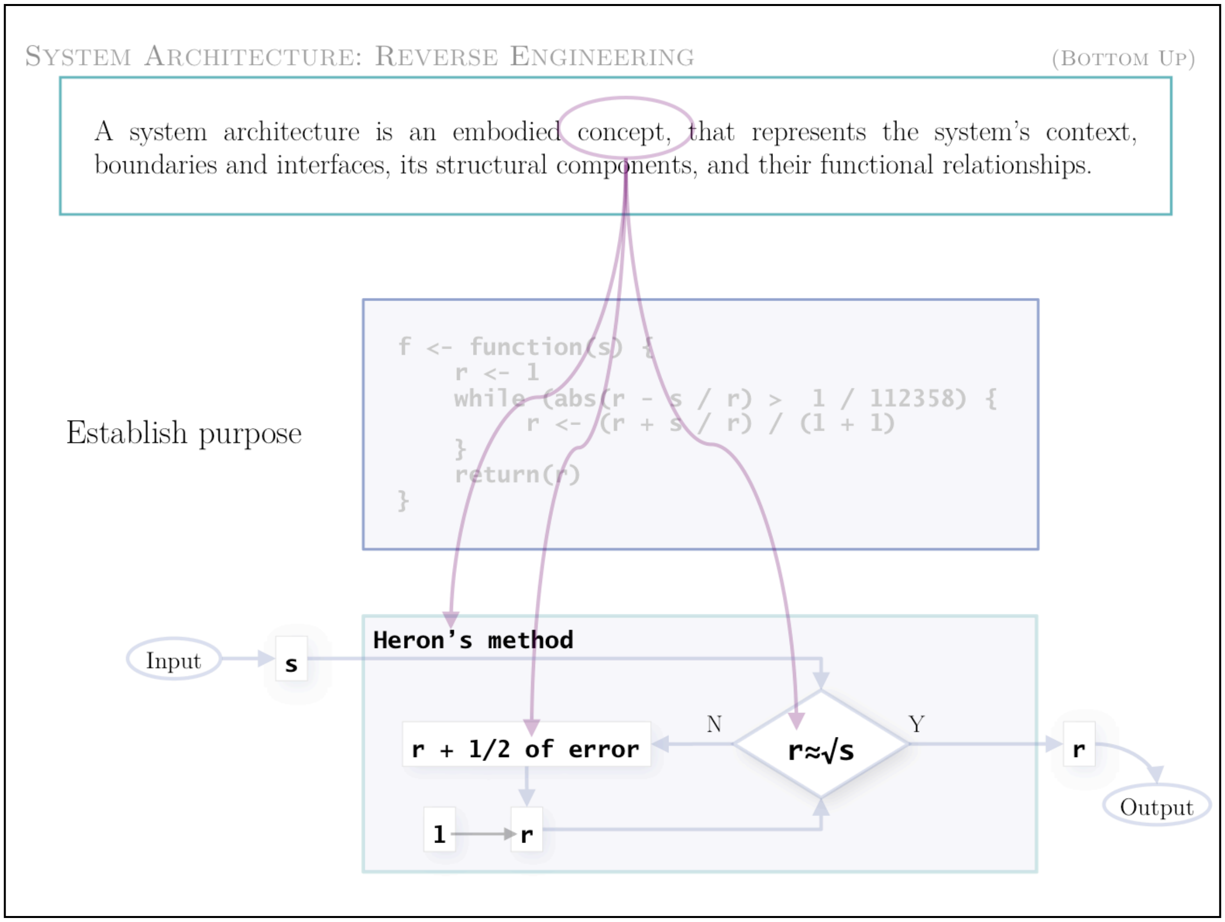
A system architecture is an embodied concept, that represents the system's context, boundaries and interfaces, its structural components, and their functional relationships.

```
f <- function(s) {
    r <- 1
    while ( abs(r - s / r) > 1 / 112358 ) {
        r <- (r + s / r) / (1 + 1)
    }
    return(r)
}
```

Structural components can be grouped to higher order units and the concept / purpose / meaning of their existence made explicit. Often the purpose / role of these modules is not obvious from the constituting components, it may be *emergent* behaviour. This grouping components and associating them with concepts is the crucial, non-automatable step in defining the architecture. This is where *real* intelligence is (still) required.

We note that 1/112358 is simply a small number – not very different from zero. From this we deduce that the while-condition asks whether the difference between r and s/r is small. This difference is the smallest when r is (nearly) equal to s/r – or, rearranged, when $s \approx r^2$. Therefore the while-condition terminates (and the function returns r) when r is close or equal to the square-root of s.

How is this achieved? This happens in the body of the while-loop. We start by setting r to 1 – which for this purpose is an arbitrary number. Then in each iteration we replace r with the mean of r and s/r. Remember: $s/r = r$ is simply another way to express: $r = \sqrt{s}$. Thus, at every iteration, r is made a little more similar to the desired quantity until it is similar enough.



System Architecture: Reverse Engineering (Bottom Up)

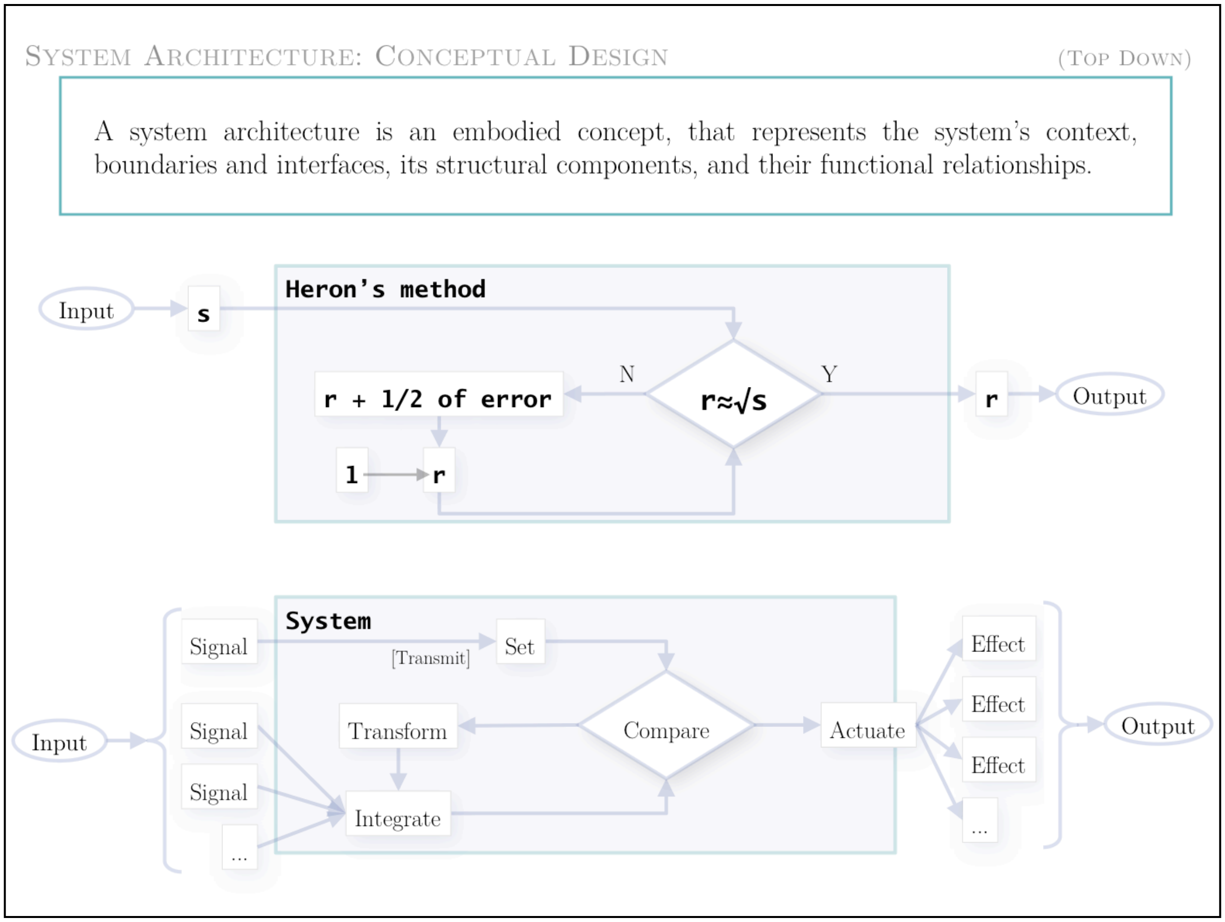
A system architecture is an embodied concept, that represents the system's context, boundaries and interfaces, its structural components, and their functional relationships.

Establish purpose

```
f <- function(s) {
    r <- 1
    while (abs(r - s / r) >  1 / 112358) {
        r <- (r + s / r) / (1 + 1)
    }
    return(r)
}
```

Once the concepts have been made explicit, we can sketch the architecture of the system in a way that is much more intuitive than the enumeration of components.

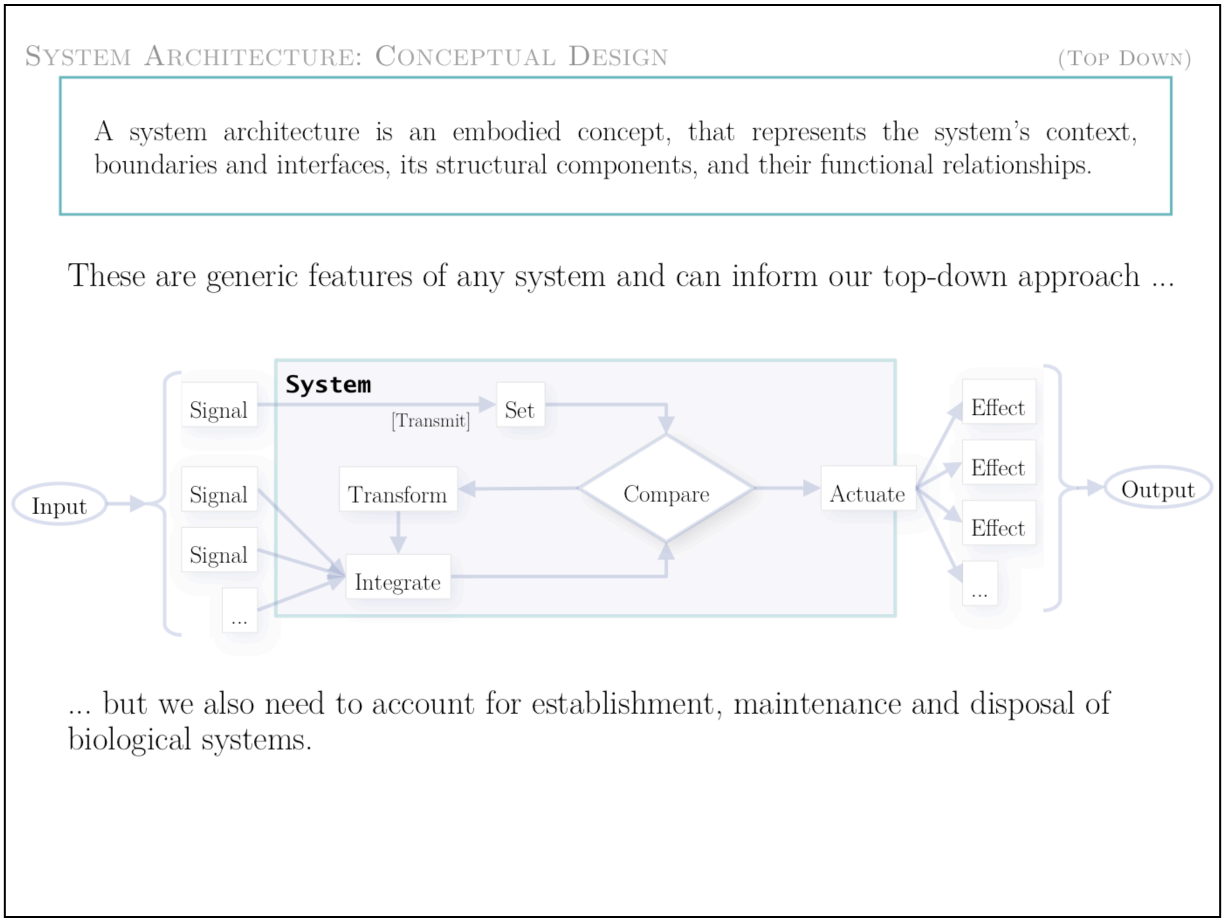
This algorithm to compute the square root of a number goes back to Heron of Alexandria (10 CE to 70 CE), i.e. it is 2,000 years old. Its time complexity is O(log(log(number/desired error)) – this means it is actually a very good algorithm.



System Architecture: Conceptual Design (Top Down)

A system architecture is an embodied concept, that represents the system's context, boundaries and interfaces, its structural components, and their functional relationships.

We can abstract the system architecture of our square-root function to general principles that apply to all systems that process information under feedback control: Input is integrated, compared against (adjustable) setpoints, transformed in a feedback loop, and ultimately actuates effects that comprise the output of the system.



System Architecture: Conceptual Design

(Top Down)

> A system architecture is an embodied concept, that represents the system’s context, boundaries and interfaces, its structural components, and their functional relationships.

These are generic features of any system and can inform our top-down approach ...

... but we also need to account for establishment, maintenance and disposal of biological systems.

How do we put this approach into practice?

The G1/s transition switch is a comparatively well understood ystem, but I hope you understand now that our purely data-driven, bottom-up approaches have significant limitations. Be reminded that most of the systems core components are not even essential genes! Obviously, the situation for many other systems for which we don’t have good primary data is even more difficult.

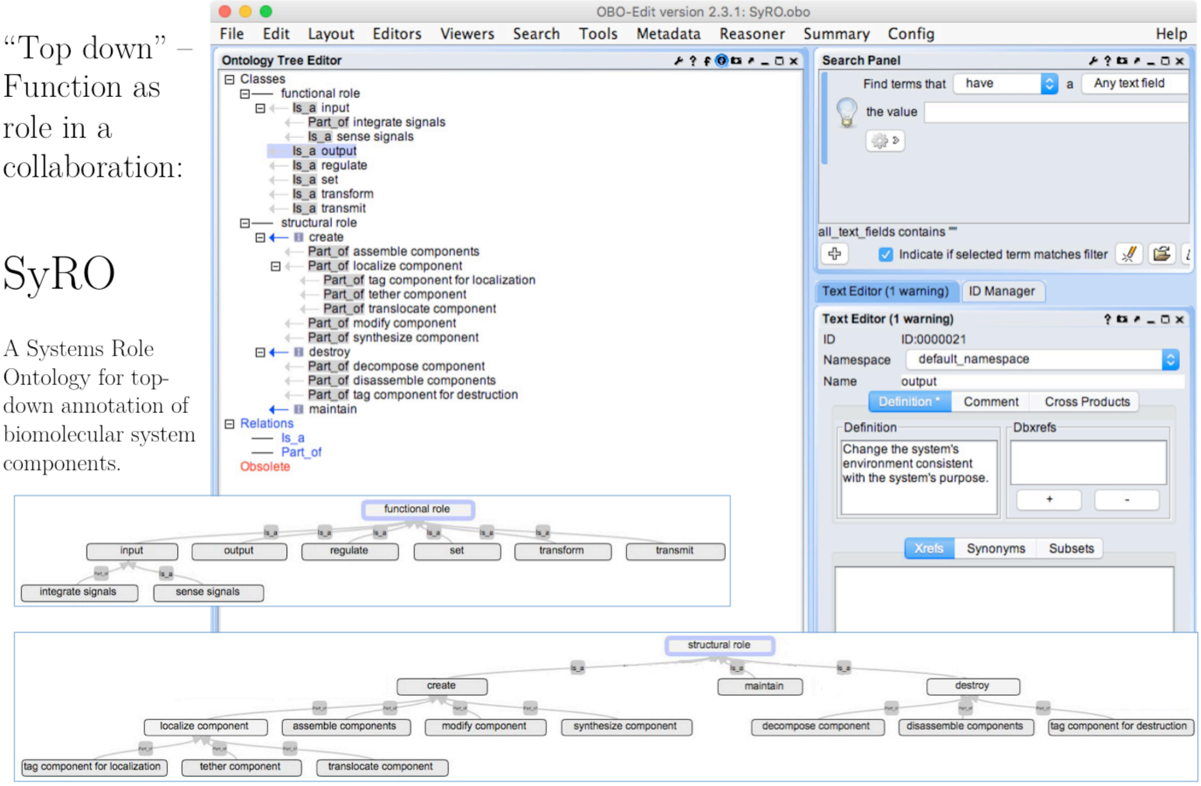
Pursuing a top-down approach is our opportunity to paint a “big picture” first, then asking in more and more detail how the logically required roles are realized by molecular components. We can include aspects of the system that we logically expect to be there, even if we have no data (yet) that they are, we are not distracted by inaccurate annotations of components – these can easily be corrected once more data is available – nor by the fact that many proteins have more than one role, and we can work through a checklist of features that are often overlooked as important system components: the roles that establish, maintain and dispose of the system.

System Architecture: Conceptual Design (Top Down)

"Top down" – Function as role in a collaboration:

SyRO

A Systems Role Ontology for top-down annotation of biomolecular system components.



```
Classes
  functional role
    Is_a input
      Part_of integrate signals
      Is_a sense signals
    Is_a output
    Is_a regulate
    Is_a set
    Is_a transform
    Is_a transmit
  structural role
    create
      Part_of assemble components
      Part_of localize component
        Part_of tag component for localization
        Part_of tether component
        Part_of translocate component
      Part_of modify component
      Part_of synthesize component
    destroy
      Part_of decompose component
      Part_of disassemble components
      Part_of tag component for destruction
    maintain
Relations
  Is_a
  Part_of
Obsolete
```

Text Editor (1 warning)

ID: ID:0000021

Namespace: default_namespace

Name: output

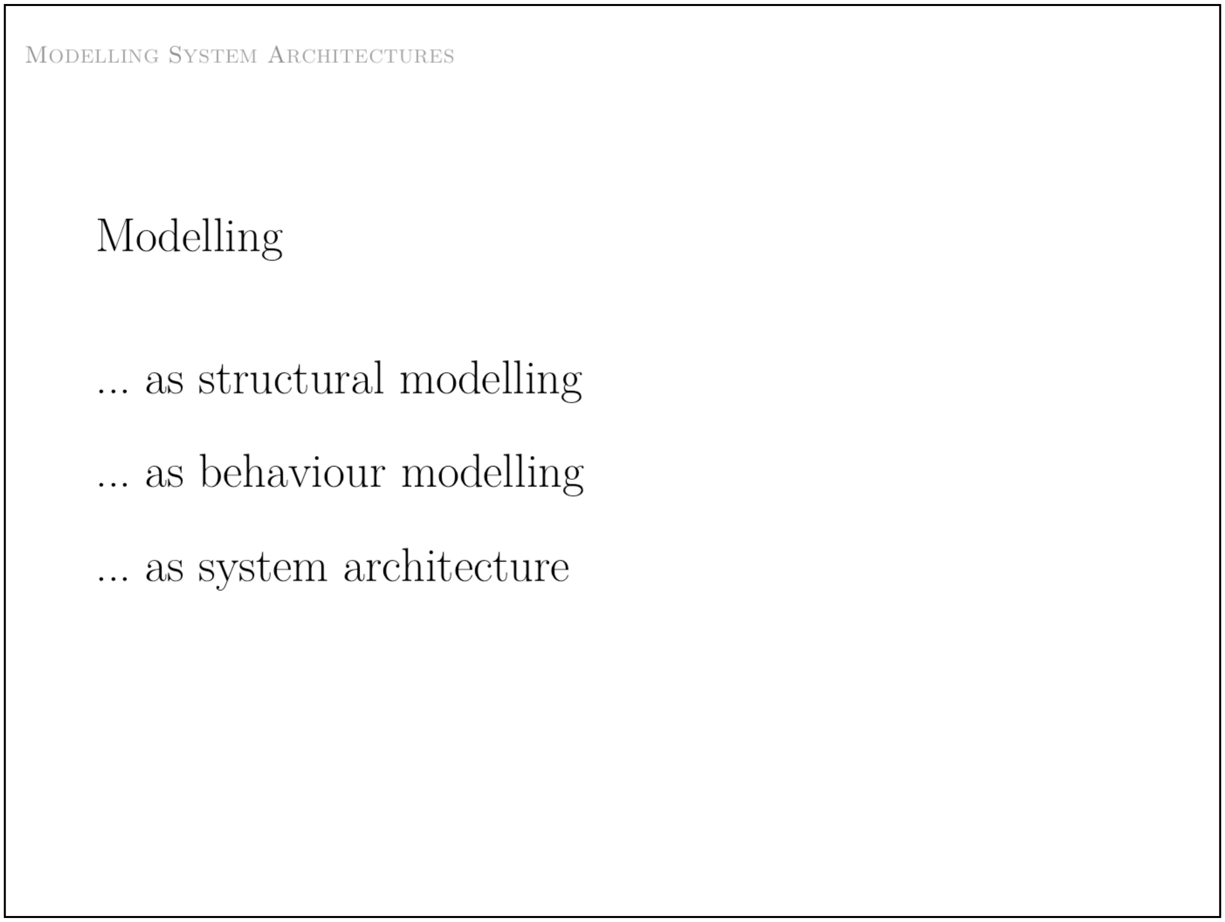
Definition: Change the system's environment consistent with the system's purpose.

Dbxrefs:

A Systems Role Ontology that includes general, abstract structural roles to create, maintain and destroy a biological system is is available as an OBO (Open Biology Ontologies) compliant file on github:

https://github.com/hyginn/SyRO

It can be downloaded, and visualized with the OBO-edit ontology editor (http://oboedit.org/).



Modelling System Architectures

Modelling

... as structural modelling

... as behaviour modelling

... as system architecture

Such systems architectures provide the vocabulary and syntax of a “language” to represent the why of biological facts. Expressing the facts in this language is a modelling task. In general engineers tend to distinguish between structural and behavioural modelling. For our purposes of biological systems architecture that may not be such a useful distinction, but let’s look at how modeling of the G1/S transition switch according to those two categories canplay out in practice.

I develop the model according to two standard drawing conventions of the UML (Unified Modeling Language), a standard of software engineering that is widely used to support software system design, an activity diagram and a structure diagram. Finally I develop the system according to a less formal system architecture paradigm.

(Disclaimer: while many of the ideas behind UML are sound, I find their actual notational conventions too often poorly designed: many UML models are overloaded with graphical fluff, have insufficient information, and include crucial notational conventions that are not explicitly spelled out in legends and that make aspects of the models obscure. But they represent the standard in the field. I am not actually endorsing UML, and I have not followed the UML conventions religiously, but you can read more about them and find more diagram types in the two links below.)

https://en.wikipedia.org/wiki/Unified_Modeling_Language

http://www.uml-diagrams.org/

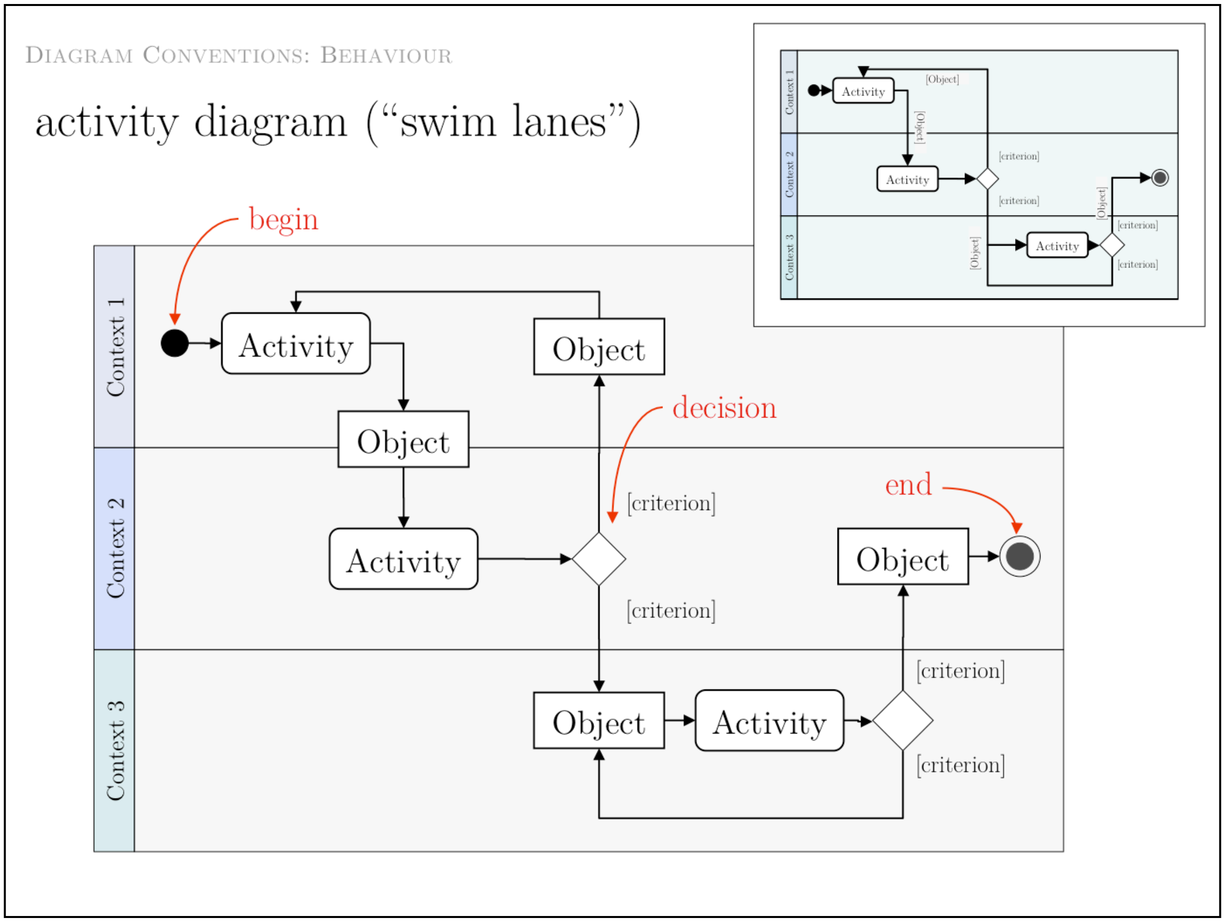


Diagram Conventions: Behaviour

activity diagram (“swim lanes”)

UML acivity diagrams associate objects and activities with particular contexts – these could be subsystems, use-cases, or actors. In principle, these are *bipartite graphs* of objects and activities, this means an object can only be joined to another object via an activity and vice versa. Edges (arrows) are directed and imply that an activity has an object as output or as input. In principle, such graphs could be created without loss of information by showing only objects, or only activities, and labelling the edges with the other categories (cf. inset). Whether this works well in practice depends on the actual system that is being modelled. Start and end of the activity are represented by black circles. The diamonds are decision points and the criteria for pursuing the activity flow in a particular direction are indicated.

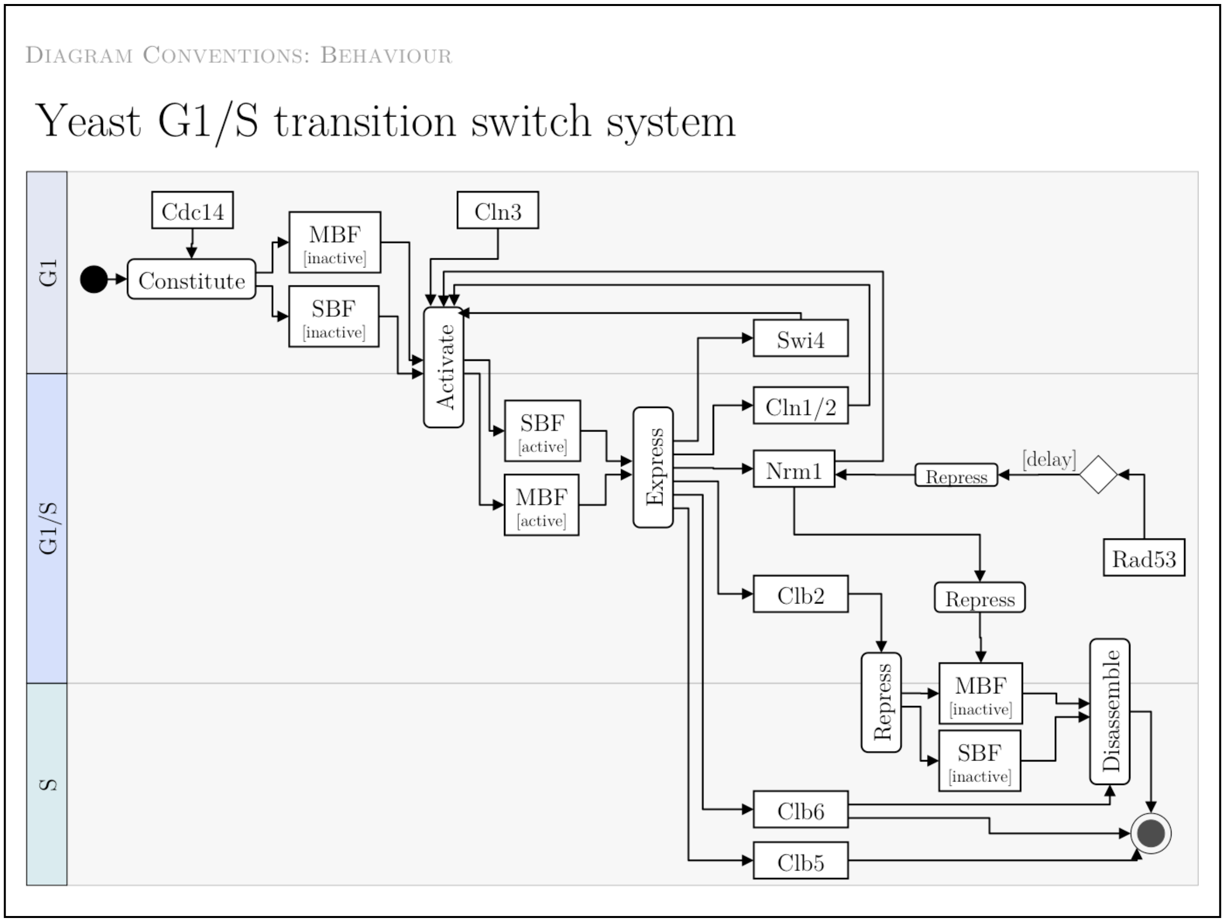


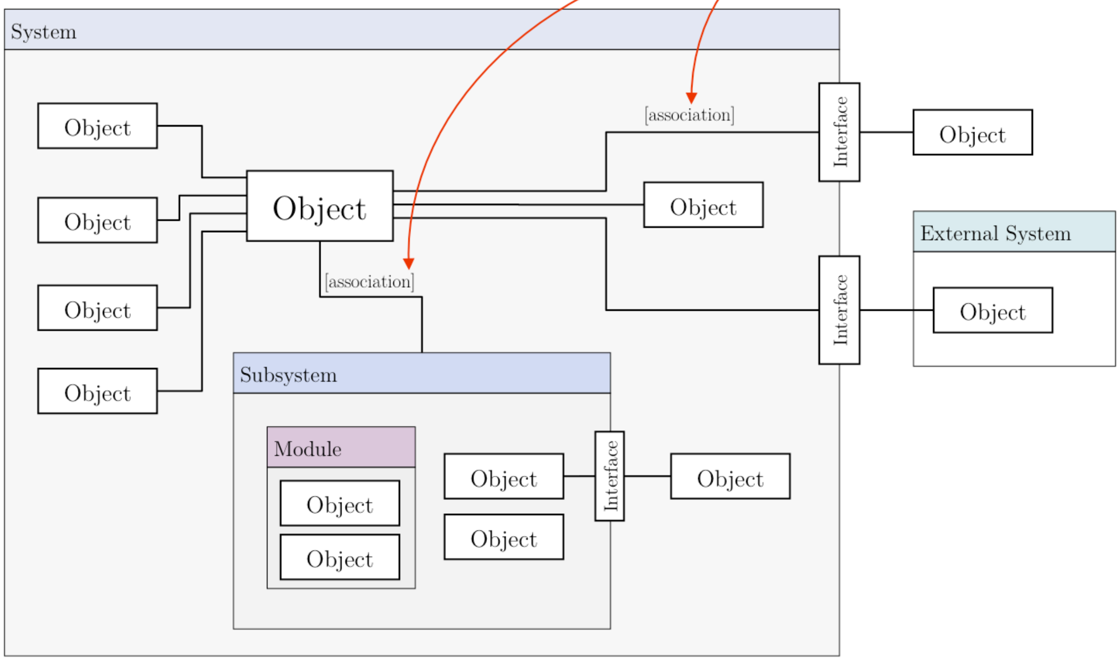
Diagram Conventions: Behaviour

Yeast G1/S transition switch system

The yeast G1/S transition switch system as an activity diagram. Gene products are “objects” and functional roles are “activities”.

DIAGRAM CONVENTIONS: STRUCTURE

structure diagram

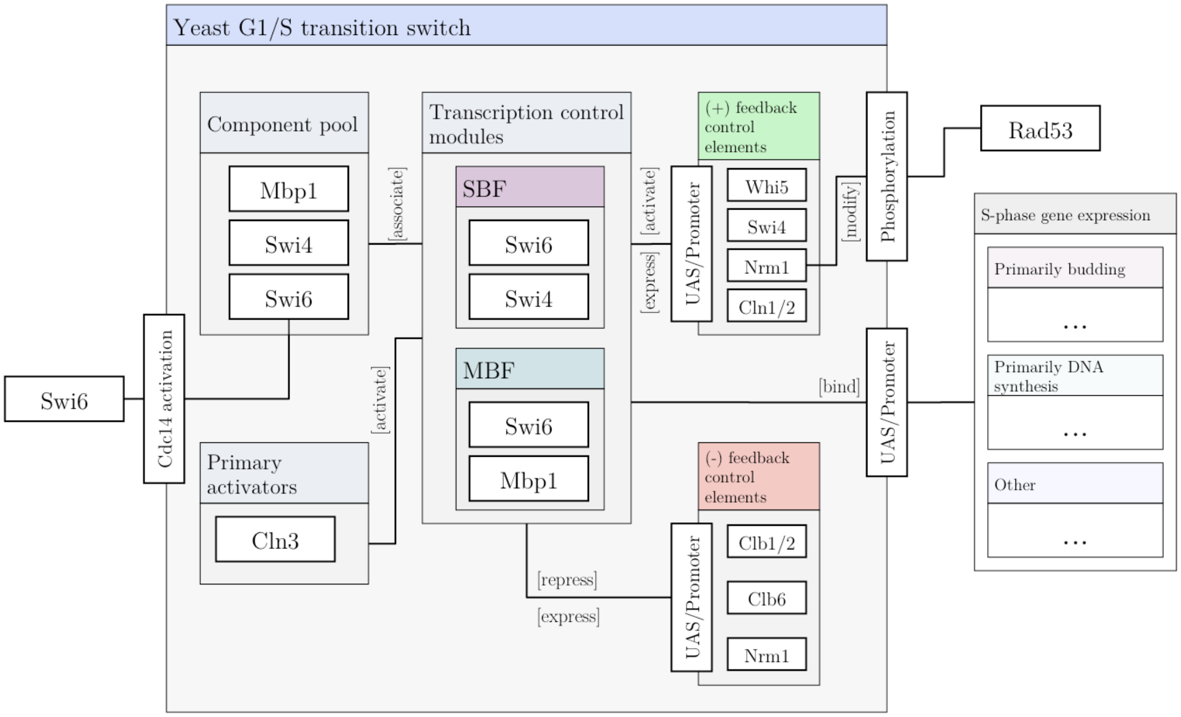


typed
association

UML structure diagrams enumerate the system's components and place them into their context of association. Function is not explicitly addressed, but particular roles can be indicated as the labels of modules or subssytems. In general associations are not typed (but they can be) and they are undirected (lines, not arrows) since an association is a symmetric relationship.

(Nb. actual UML structure diagrams use an obscure distinction between actual and referenced roles, a convention of object names that indicate their status as artefact e.g. software class, hardware object and they use an odd box + cup-shape as a symbol for an interface but don't label the interface properly.)

Diagram Conventions: Structure



Yeast G1/S transition switch

Component pool

Mbp1

Swi4

Swi6

[associate]

Transcription control modules

SBF

Swi6

Swi4

MBF

Swi6

Mbp1

[activate]

[express]

UAS/Promoter

(+) feedback control elements

Whi5

Swi4

Nrm1

Cln1/2

[modify]

Phosphorylation

Rad53

Swi6

Cdc14 activation

[activate]

Primary activators

Cln3

[bind]

UAS/Promoter

S-phase gene expression

Primarily budding

...

Primarily DNA synthesis

...

Other

...

(-) feedback control elements

UAS/Promoter

Clb1/2

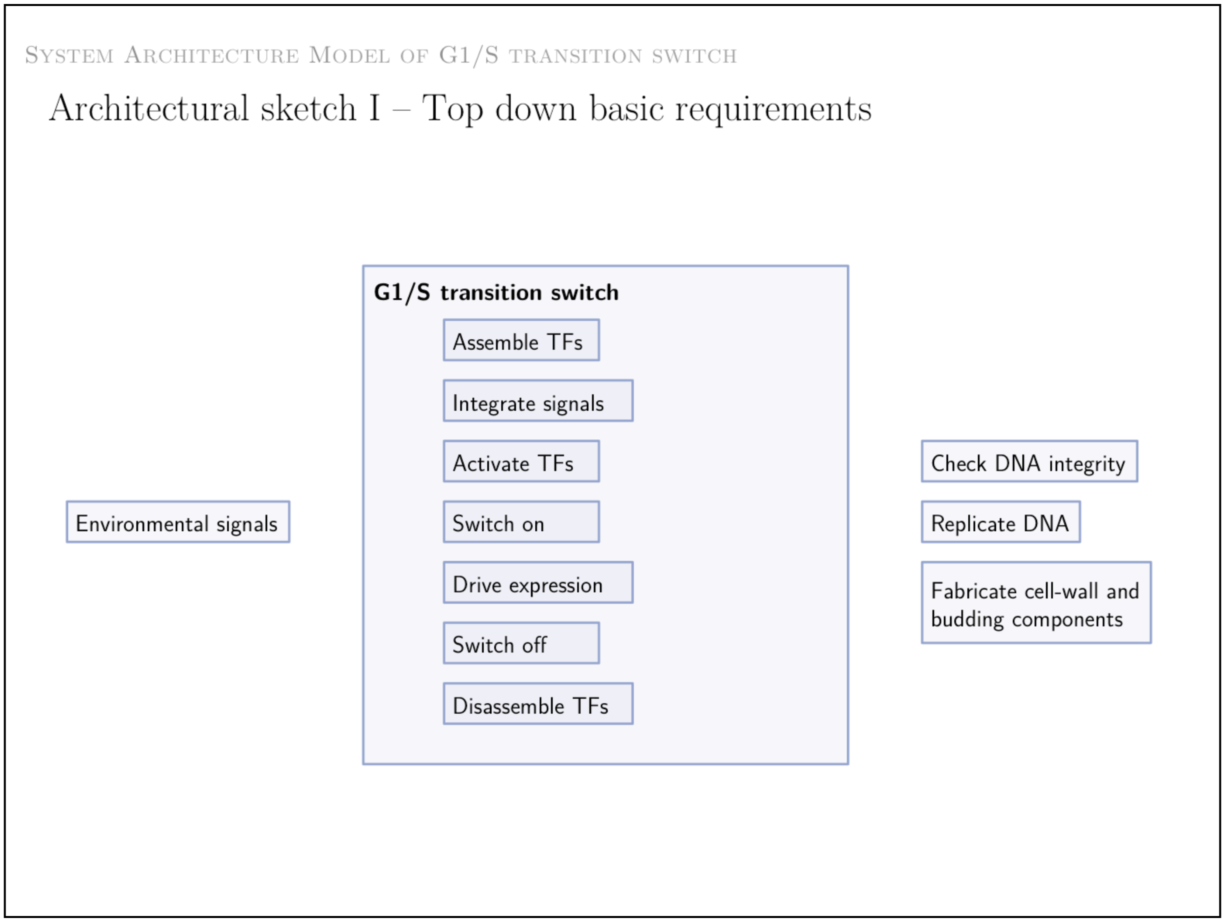
Clb6

Nrm1

[repress]

[express]

The yeast G1/S transition switch system as a structure diagram. Gene products are white boxes, modules and subsystems are explicitly labelled, as are the activities that interface extenal system. Note that this structural display gives us an opportunity to emphasize the crucial and separate roles of (+) and (-) – feedback control components.



System Architecture Model of G1/S transition switch

Architectural sketch I – Top down basic requirements

Environmental signals

G1/S transition switch

Assemble TFs

Integrate signals

Activate TFs

Switch on

Drive expression

Switch off

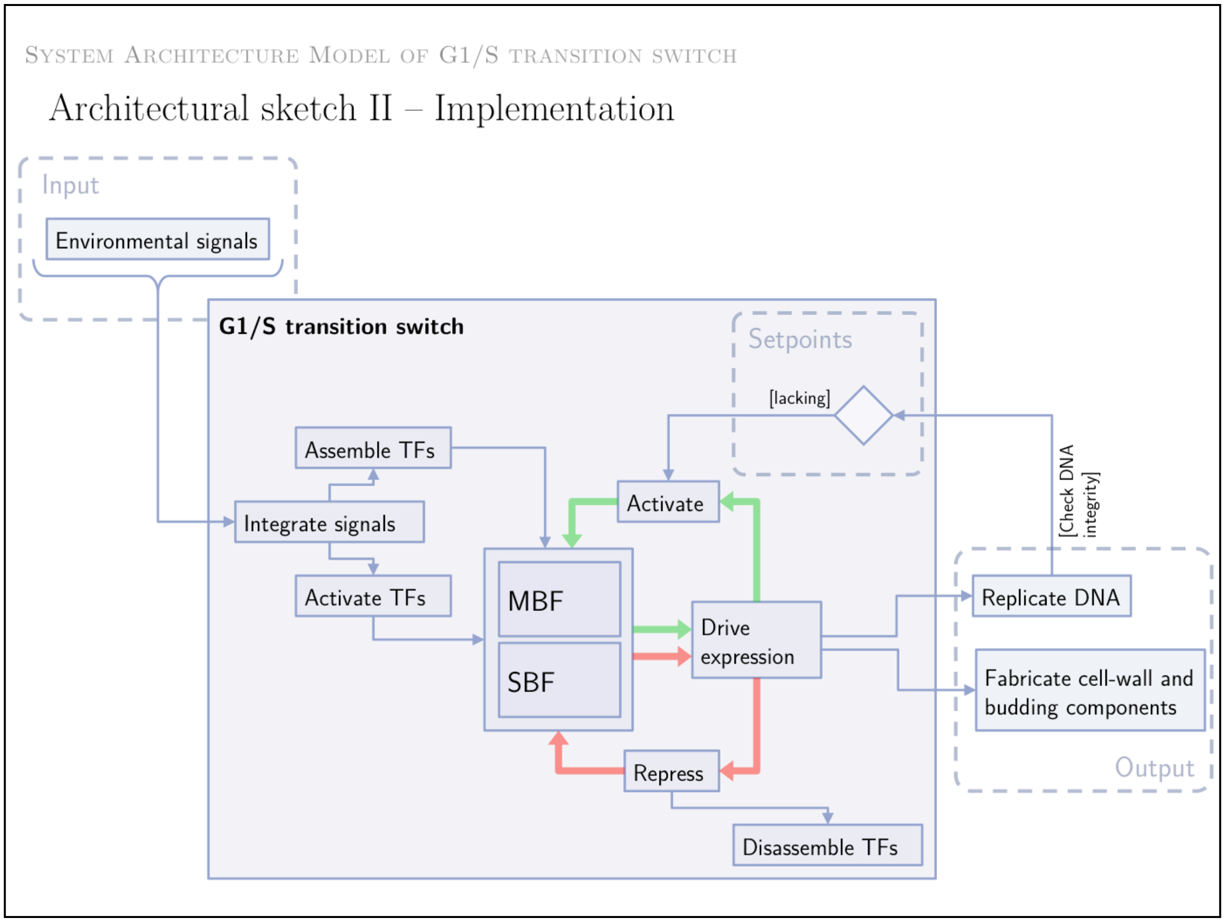
Disassemble TFs

Check DNA integrity

Replicate DNA

Fabricate cell-wall and budding components

An architectural model integrates function and behaviour, although not in the bottom-up way that we have looked at previously, but guided by the salient conceptual features of the system. Rather than reverse engineering, we may start from forward engineering a list of requirements. This is what we may postulate about the system's abilities *from first principles*.



System Architecture Model of G1/S transition switch

Architectural sketch II – Implementation

Input

Environmental signals

G1/S transition switch

Setpoints

[lacking]

Assemble TFs

Integrate signals

Activate TFs

Activate

MBF

SBF

Drive expression

Repress

Disassemble TFs

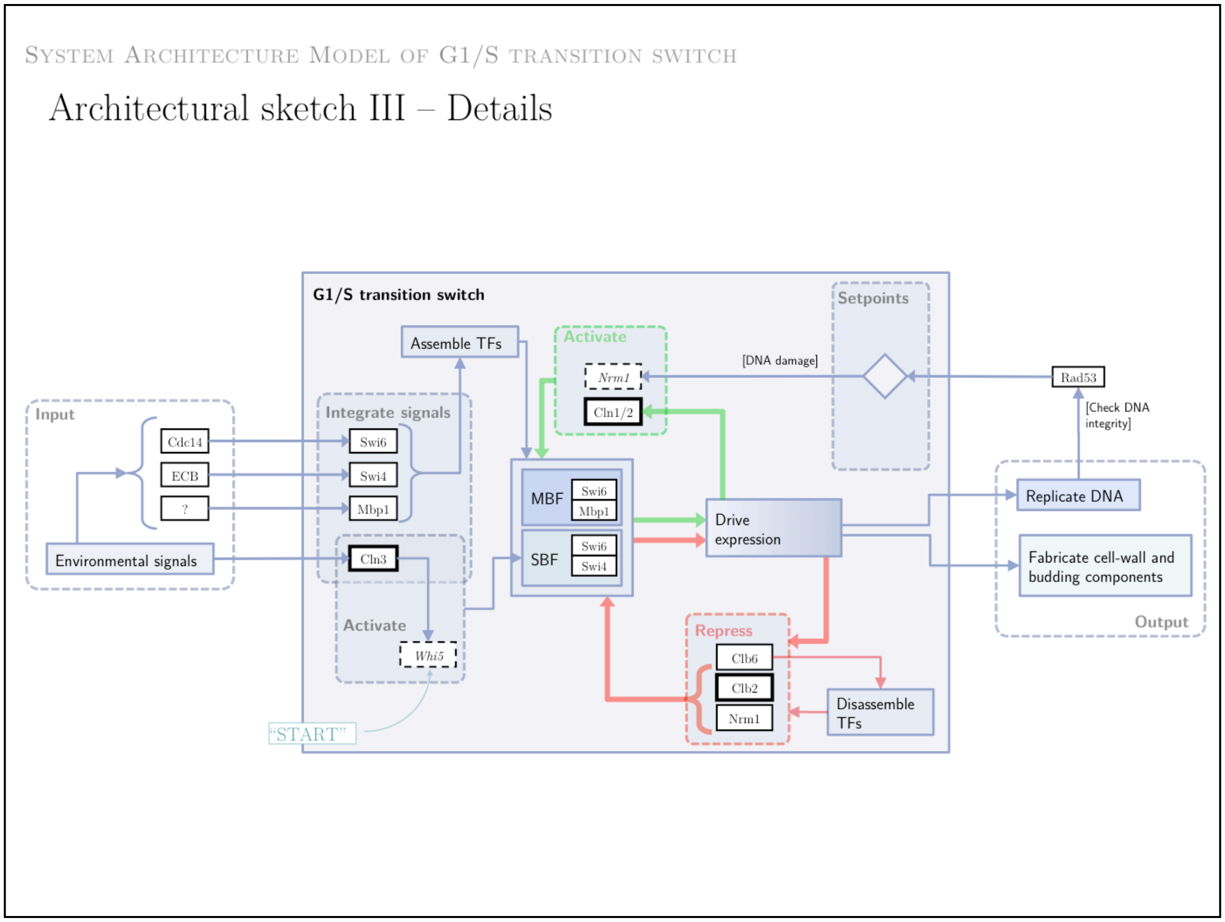
[Check DNA integrity]

Replicate DNA

Fabricate cell-wall and budding components

Output

As we implement the model, we take care that all of our requirements are explicitly represented. We may rename the requirements from their original abstraction, if their implementation is sufficiently concrete. Basic systems components like input, output, setpoints can be made explicit.



System Architecture Model of G1/S transition switch

Architectural sketch III – Details

Once the basic map has been created, we can add details to whatever depth we desire. If necessary, rather than overload the sketch, we could define subsystems and detail these in a separate sketch. The only requirement is that the logic of connections between elements should map between the different sketches – i.e. the more detailed sketch should be a drop-in replacement of the more general one.

In this representation of the G1/S cell cycle switch I have colored the two inner feedback loops of the system green and red to emphasize the implementation of the switching logic. Color is used sparingly and in an intutive way: to match MBF with DNA replication and SBF with cell-wall biosynthesis, and by adding a slight gradient to the "Drive expression" box to suggest the early/late expression of feedforward ("on"-switch) and feedback ("off"-switch) regulators. The three key cyclins that are responsible for the internal control of the switch are boxed with bold lines, the two TF inhibitors whose inactivation leads to de-repression of SBF resp. MBF, Whi5 resp. Nrm1 are boxed with broken lines.

With this sketch we have modelled a representation of the G1/S switch that accomodates the facts, makes the roles explicit, and promotes our understanding of the goals of the system.

http://steipe.biochemistry.utoronto.ca/abc

BORIS.STEIPE@UTORONTO.CA

DEPARTMENT OF BIOCHEMISTRY & DEPARTMENT OF MOLECULAR GENETICS
UNIVERSITY OF TORONTO, CANADA