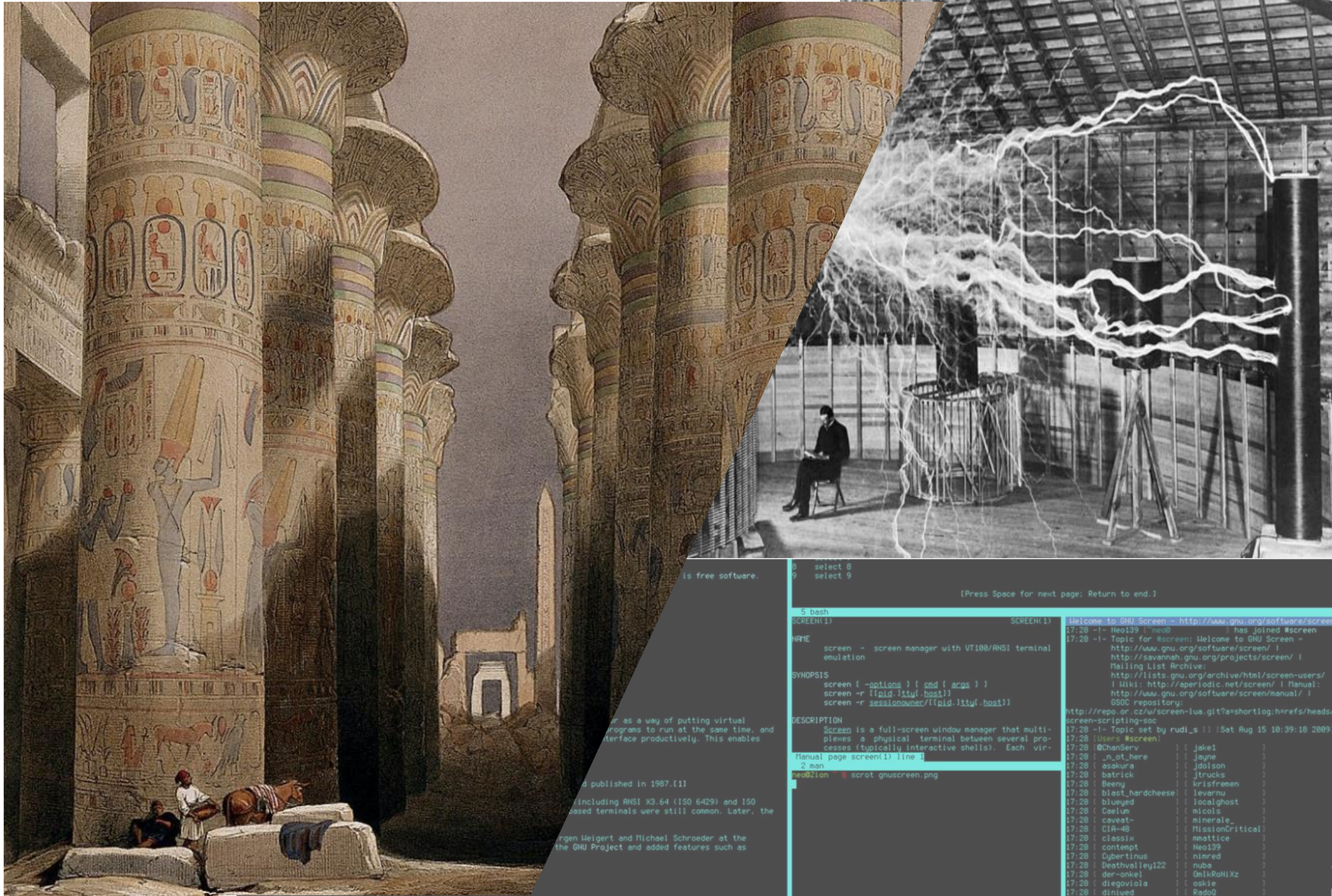
Knowledge Representation

Systems Design and Analysis

IT.304 Systems analysis, design, and implementation planning

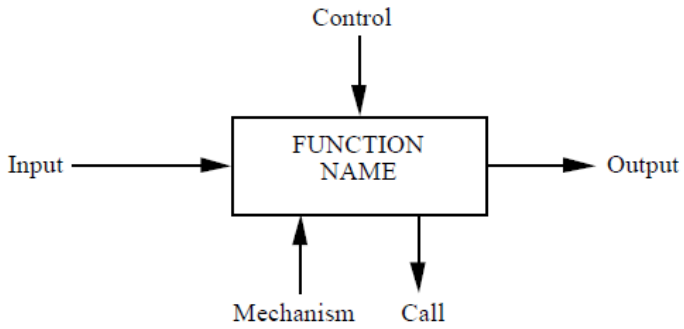
www.snhu.edu
b.hogan@snhu.edu
bh.github

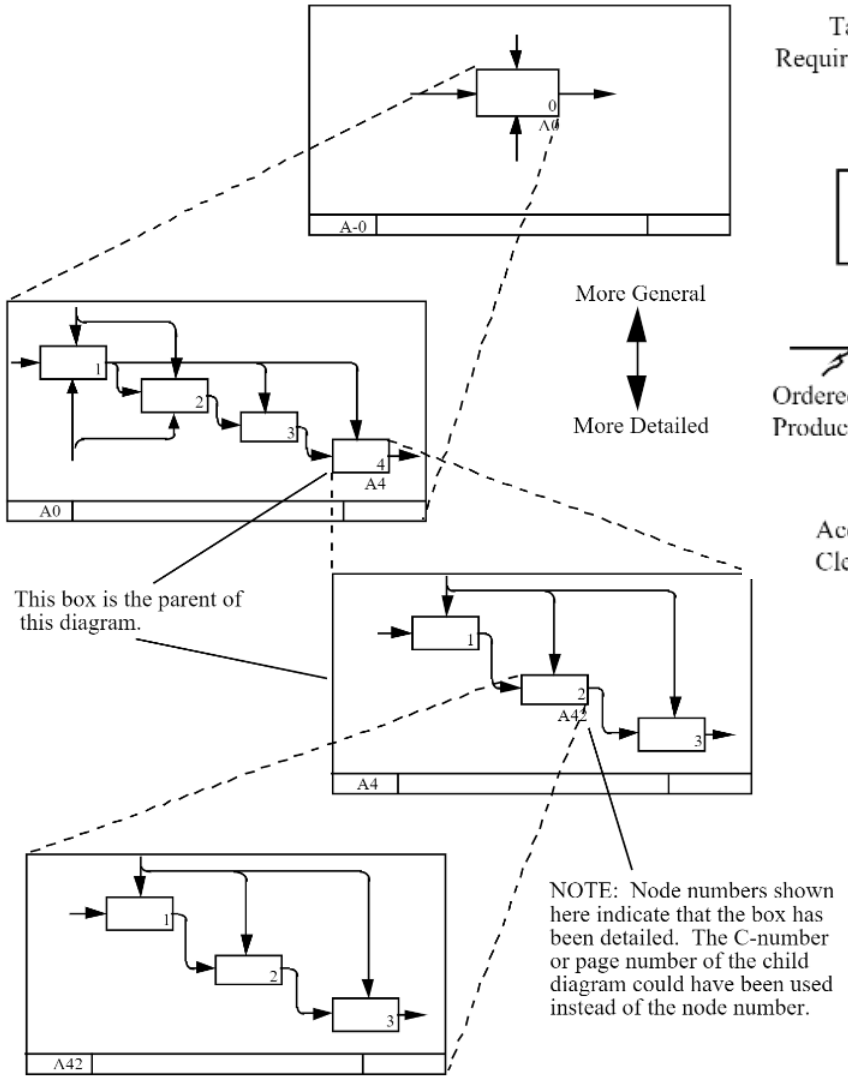


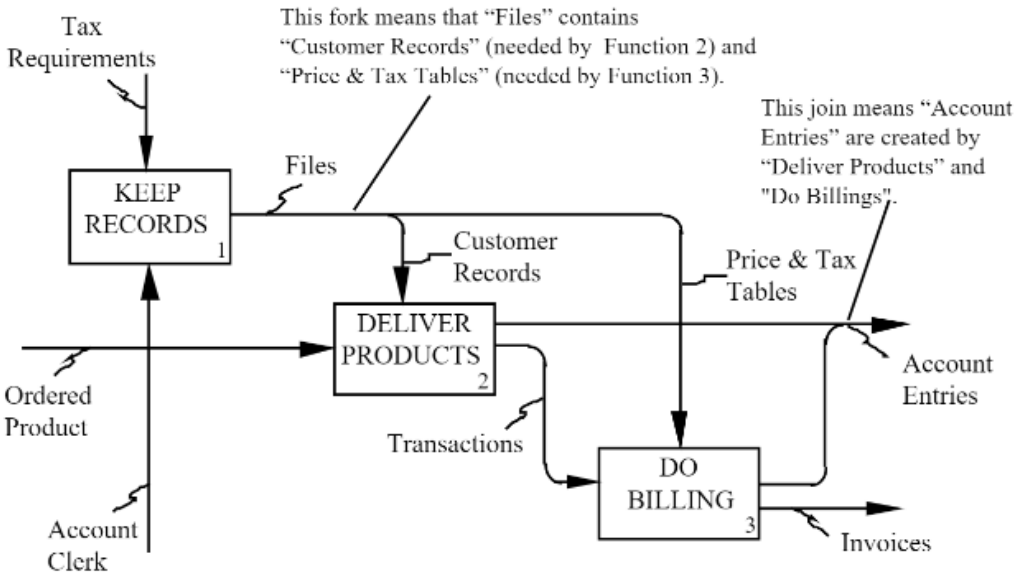
Model.1: IDEF0 <bh.github> <website> <how.to.doc> <how.to.video> <wikipedia>

History: developed in 1970s by U.S. Air Force for improving manufacturing of fighting aircraft.

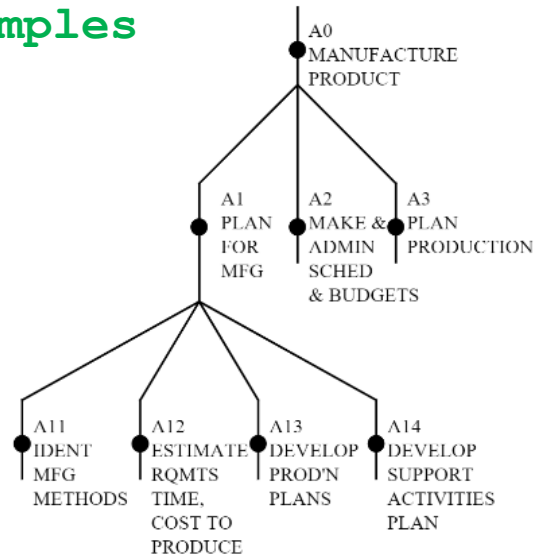
Purpose: gather process inputs, outputs, resources, and constraints to methodically outline a system or specific process.







Examples



Model.2: SWOT analysis <bh.github> <website> <how.to.video> <wikipedia>

History: deep history and university dispute on ownership

Purpose: Apply this versatile model to personal situations and businesses to perform a high-level assessment of the value of performing system analysis work. It is also instrumental in helping to organize an unorganized group discussion quickly.

INTERNAL	**STRENGTHS** Capabilities, resources or attributes that provide a competitive advantage Examples: production capacity, industry experience, financial resources, unique ingredients or packaging	**WEAKNESSES** Capabilities, resources or attributes that need improvement Examples: lack of resources, limited experience, no marketing plan, no food safety or traceability program
EXTERNAL	**OPPORTUNITIES** Circumstances that if capitalized on could have a positive impact on the business Examples: favourable market trends, new technology, government policy changes, potential partnerships	**THREATS** Circumstances that do or could have a negative impact on the business Examples: regulatory changes; new trends, access to ingredients, exchange rate fluctuations

EXAMPLE SWOT ANALYSIS

STRENGTHS	WEAKNESSES
Company: • Food science expertise • Sales have increased 57% in past six months • Low rent due to location (Tofino) • Utilizing low cost, approved, gluten-free, shared commercial kitchen* **Product:** • Contains less sugar and more protein than competitive products • Is on-trend (gluten-free, locally made) • Is listed in local food stores	**Company:** • Lack of financing • Has no business or marketing plan • High distribution cost due to location • Utilizing shared commercial kitchen limits production capacity* **Product:** • Higher priced than competitive products • High ingredient costs • Three month shelf life insufficient for retail sales
OPPORTUNITIES	**THREATS**
Company: • High number of people with gluten sensitivities on Vancouver Island • A higher volume in online purchases of natural health food products all over British Columbia	**Company:** • CFIA's Food Labelling Modernization Initiative may impact the company • Trend for gluten-free on downturn • Crop failure has affected supply of the ancient grains

*Note that a strength can also be a weakness. In this example the cookies are produced in a low cost shared commercial kitchen, meeting food safety requirements and gluten-free status. While a strength for a newly established company, it is also a weakness in that production in a shared commercial kitchen limits production capacity.

Model.2: SWOT analysis <bh.github><website><how.to.doc><wikipedia>

- Note: these models are for class quick reference use only.
- All students in IT.304 purchased and own "the decision book."

THE SWOT ANALYSIS

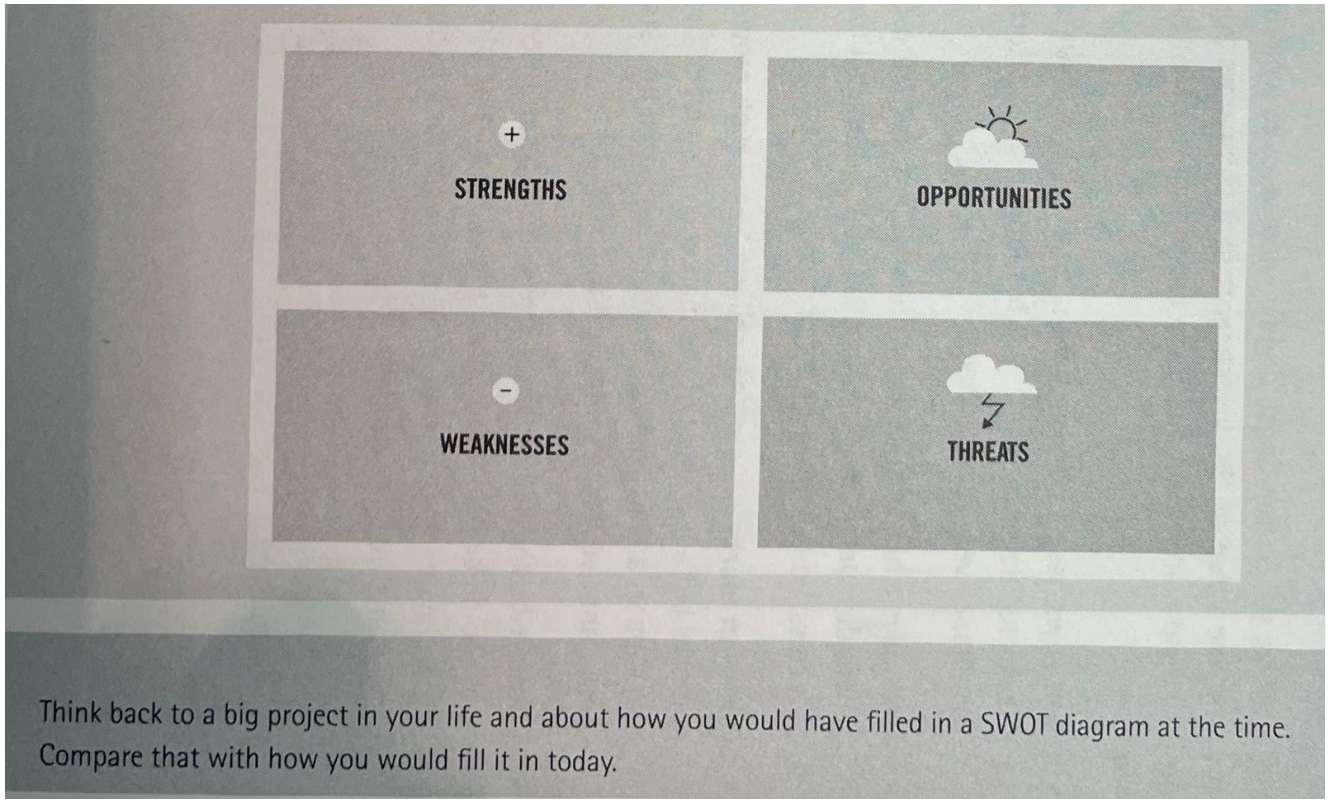
HOW TO FIND THE RIGHT SOLUTION

With SWOT analysis, you evaluate the Strengths, Weaknesses, Opportunities and Threats identified in a project. The technique is based on a Stanford University study from the 1960s which analyzed data from Fortune 500 companies. The study found a 35 percent discrepancy between the companies' objectives and what was actually implemented. The problem was not that the employees were incompetent but that the objectives were too ambiguous. Many employees didn't even know why they were doing what they were doing. SWOT was developed from the results of the study to help those involved in a project to gain a clearer understanding of it.

It is worth taking the time to think about each step of the SWOT analysis rather than just hastily fill it out. How can we emphasize our strengths and compensate for (or cover up) our weaknesses? How can we maximise opportunities? How can we protect ourselves against threats?

What is interesting about SWOT analysis is its versatility: it can be applied to business and personal decisions with equal success.

If you're not failing, you're not trying hard enough. *Gretchen Rubin*



A) Retrieved from: Krogerus, M., Tschappeler, R., and Pienning, J. (2018). **The decision book: fifty models for strategic thinking**. ISBN-10: 0393652378, ISBM-13, 978-0393652376.

- Amazon.com: The Decision Book: Fifty Models for Strategic Thinking: 9780393652376: Krogerus, Mikael, Tschäppeler, Roman, Piening, Jenny: Books

Model.3:
Swimlane diagram

<bh.github>

<website>

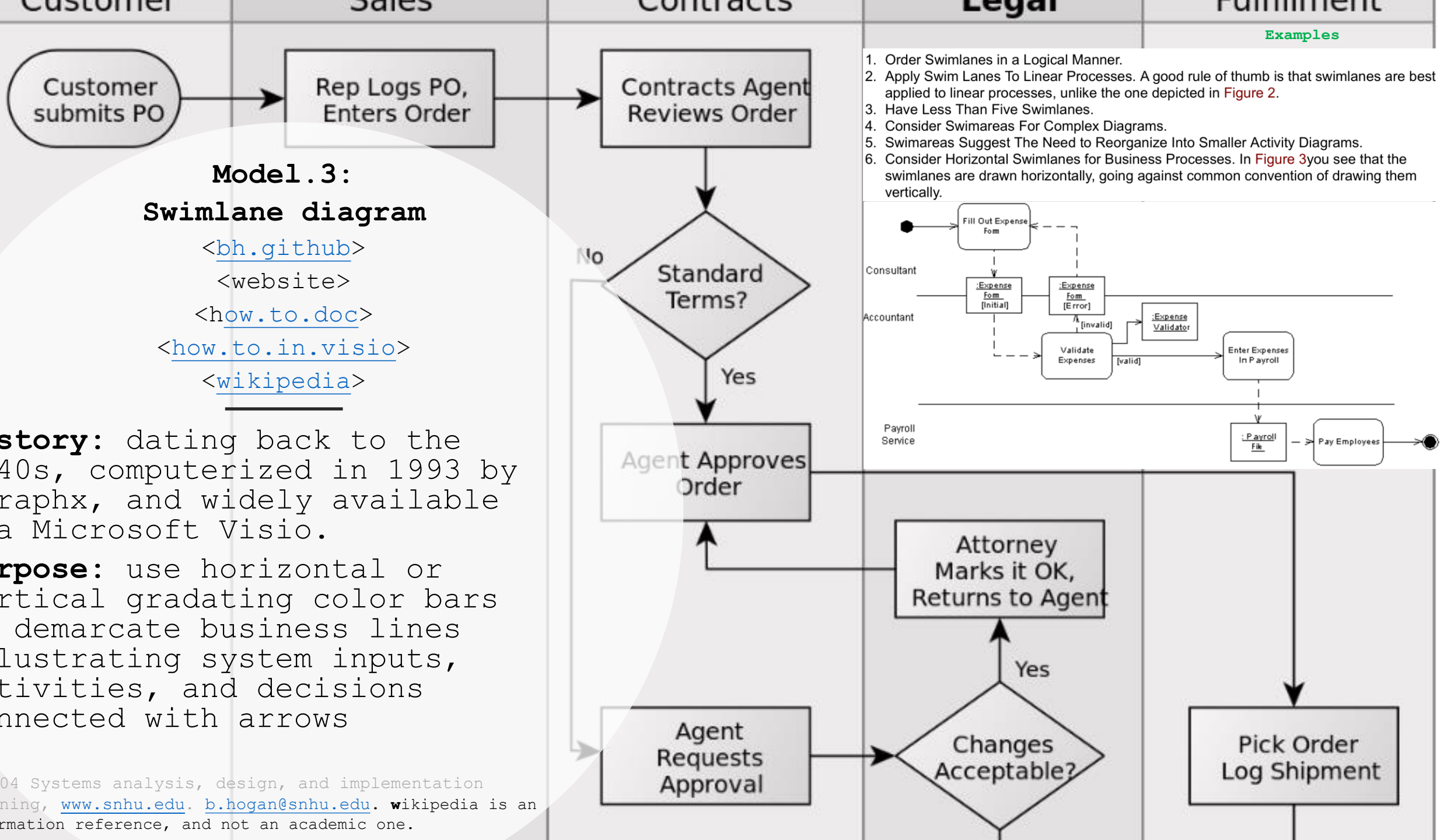
<how.to.doc>

<how.to.in.visio>

<wikipedia>

History: dating back to the 1940s, computerized in 1993 by Igraphx, and widely available via Microsoft Visio.

Purpose: use horizontal or vertical gradating color bars to demarcate business lines illustrating system inputs, activities, and decisions connected with arrows



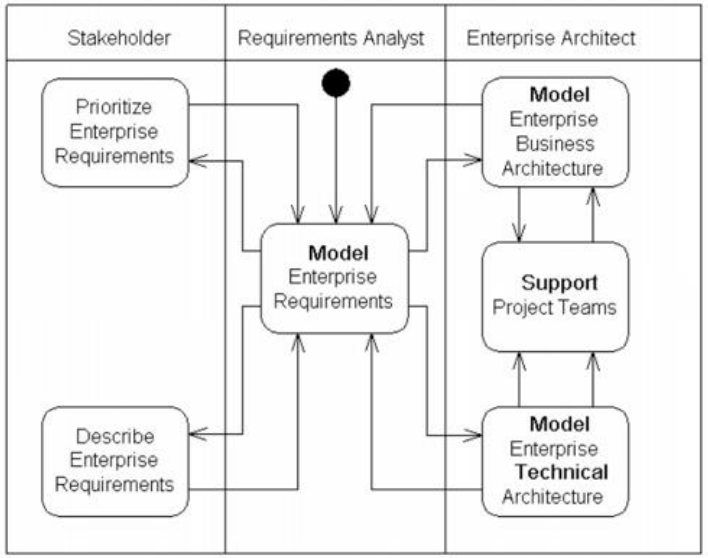
Examples

1. Order Swimlanes in a Logical Manner.
2. Apply Swim Lanes To Linear Processes. A good rule of thumb is that swimlanes are best applied to linear processes, unlike the one depicted in Figure 2.
3. Have Less Than Five Swimlanes.
4. Consider Swimareas For Complex Diagrams.
5. Swimareas Suggest The Need to Reorganize Into Smaller Activity Diagrams.
6. Consider Horizontal Swimlanes for Business Processes. In Figure 3you see that the swimlanes are drawn horizontally, going against common convention of drawing them vertically.

6. Swimlane Guidelines

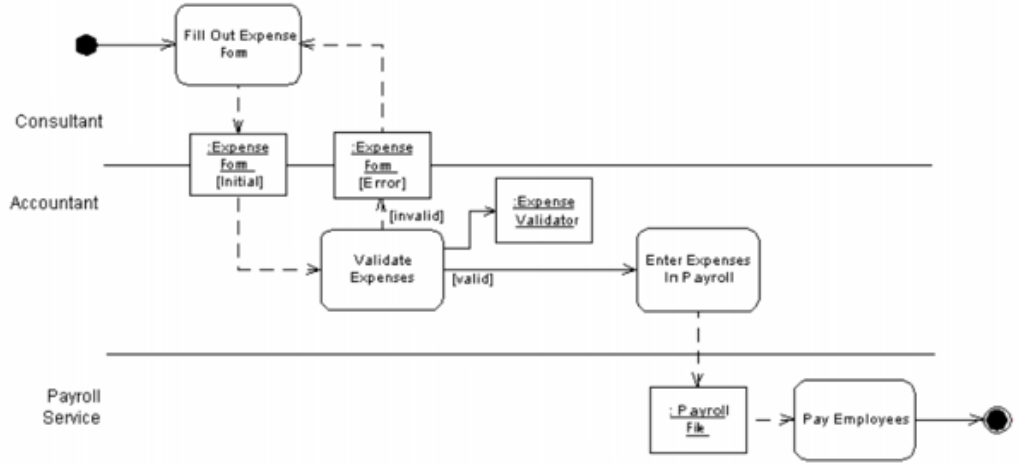
A swimlane is a way to group activities performed by the same actor on an activity diagram or to group activities in a single thread. Figure 2 includes three swimlanes, one for each actor.

Figure 2. A UML activity diagram for the enterprise architectural modeling (simplified).



http://www.agilemodeling.com/style/activityDiagram.htm#Swimlanes

Figure 3. Submitting expenses.



1. Order Swimlanes in a Logical Manner.
2. Apply Swim Lanes To Linear Processes. A good rule of thumb is that swimlanes are best applied to linear processes, unlike the one depicted in Figure 2.
3. Have Less Than Five Swimlanes.
4. Consider Swimareas For Complex Diagrams.
5. Swimareas Suggest The Need to Reorganize Into Smaller Activity Diagrams.
6. Consider Horizontal Swimlanes for Business Processes. In Figure 3you see that the swimlanes are drawn horizontally, going against common convention of drawing them vertically.

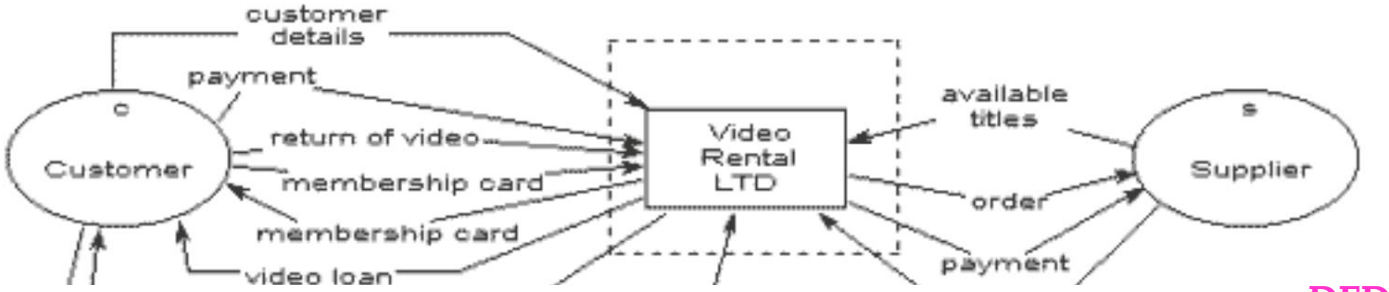
7 Action-Object Guidelines

Activities act on objects, In the strict object-oriented sense of the term an action object is a system object, a software construct. In the looser, and much more useful for business application modeling, sense of the term an action object is any sort of item. For example in Figure 3the *ExpenseForm* action object is likely a paper form.

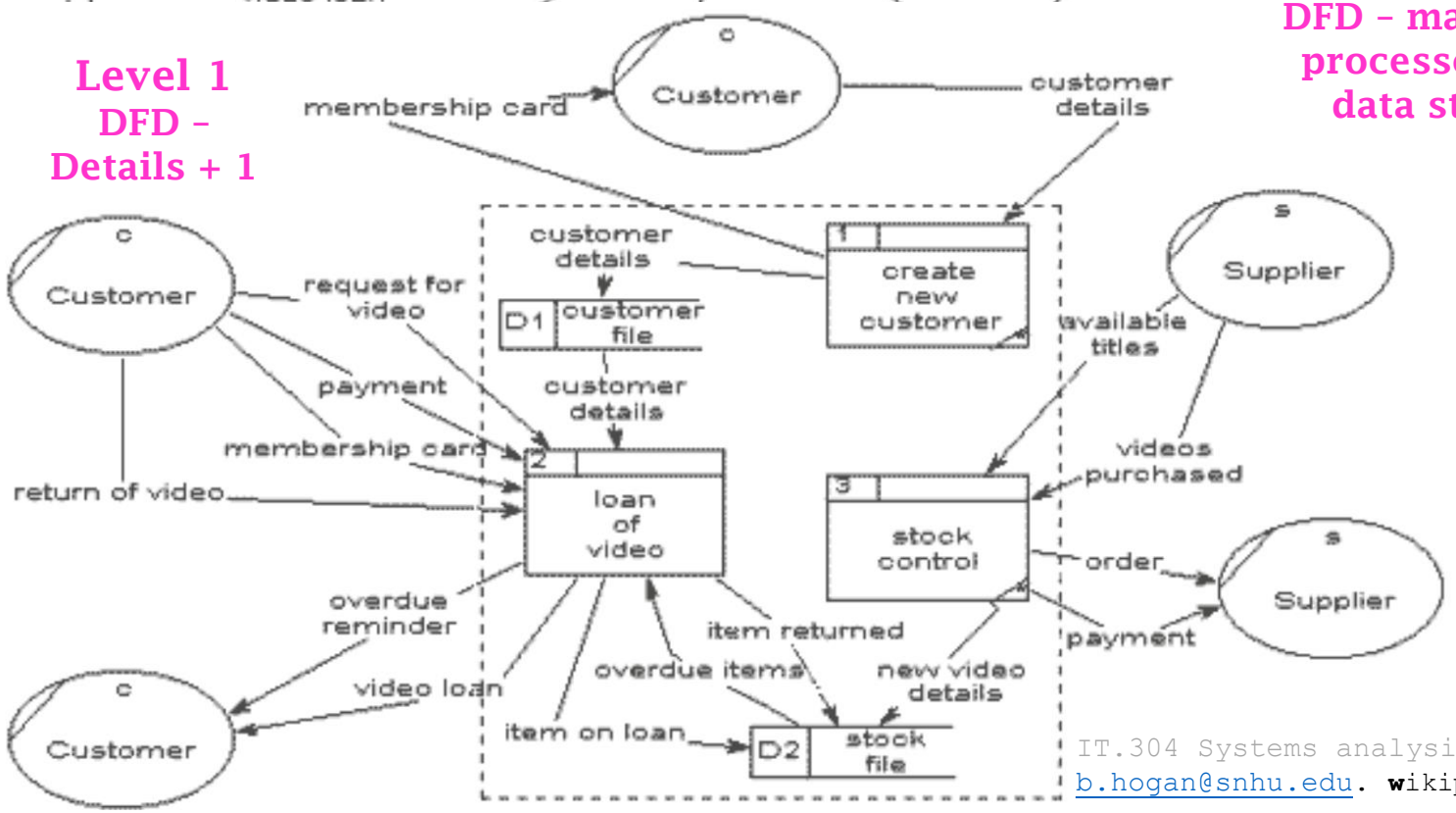
Model.4: Data Flow Diagraming <sparx-models> <website> <how.to.doc[VG]> <how.to.video> <wikipedia>

Purpose: proess of representing simplified data transactions to help process and stakeholder owners agree on scope and boundaries of a systems analysis and design reengineering effort. Level 0 is the highlevel context. Key tasks are detailed in level 1 indicating storage medium and transactions. Level 2 specifies transactions and their storage.

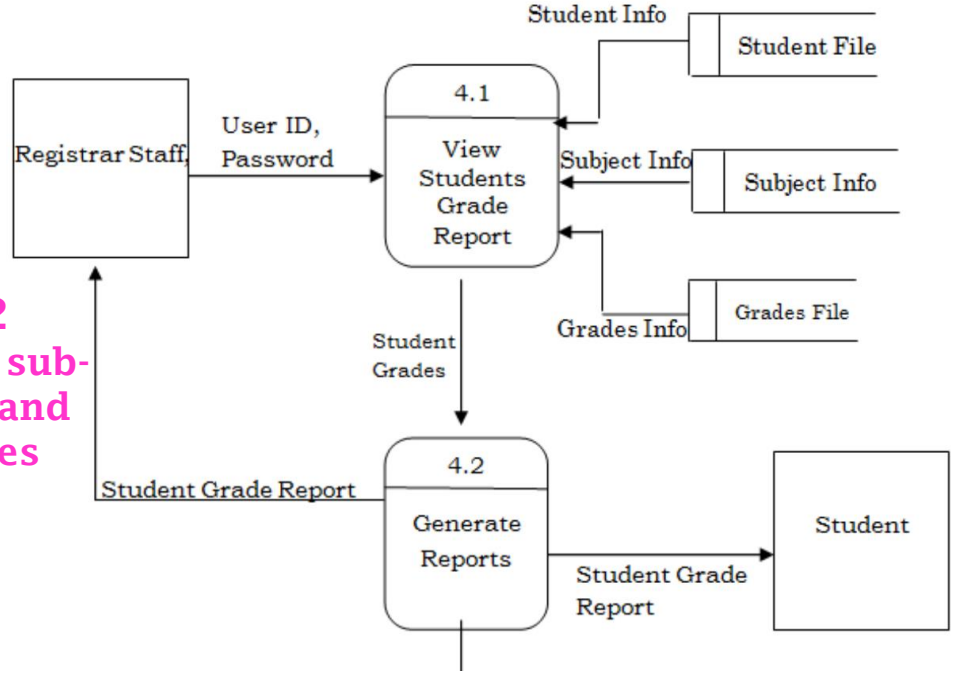
Level 0 - DFD - Context Diagram



Level 1 DFD - Details + 1



Level 2 DFD - main sub-processes and data stores



- **Context diagrams** — context diagram DFDs are diagrams that present an overview of the system and its interaction with the rest of the "world".
- **Level 1 data-flow diagrams** — present a more detailed view of the system than context diagrams, by showing the main sub-processes and stores of data that make up the system as a whole.
- **Level 2 (and lower) data-flow diagrams** — a major advantage of the data-flow modelling technique is that, through a technique called "levelling", the detailed complexity of real world systems can be managed and modeled in a hierarchy of abstractions. Certain elements of any dataflow diagram can be decomposed ("exploded") into a more detailed model a level lower in the hierarchy.

Model.5 Data Flow Diagram combined with data table normalization:<bh.github>

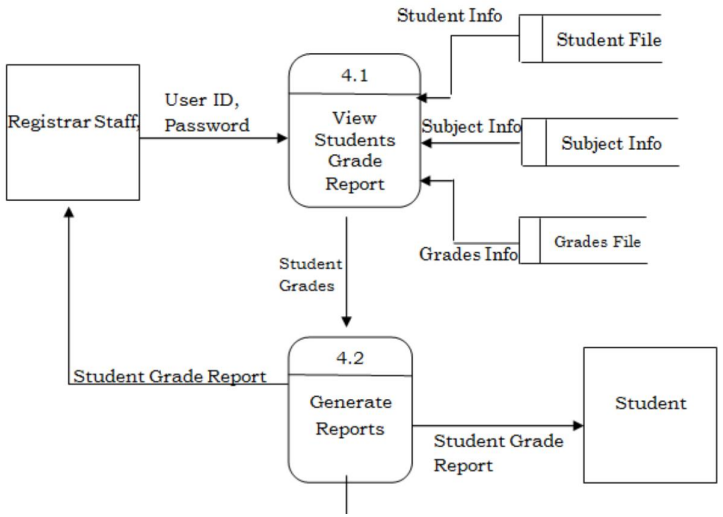
History: Purpose: <draft>

relational

database

tables

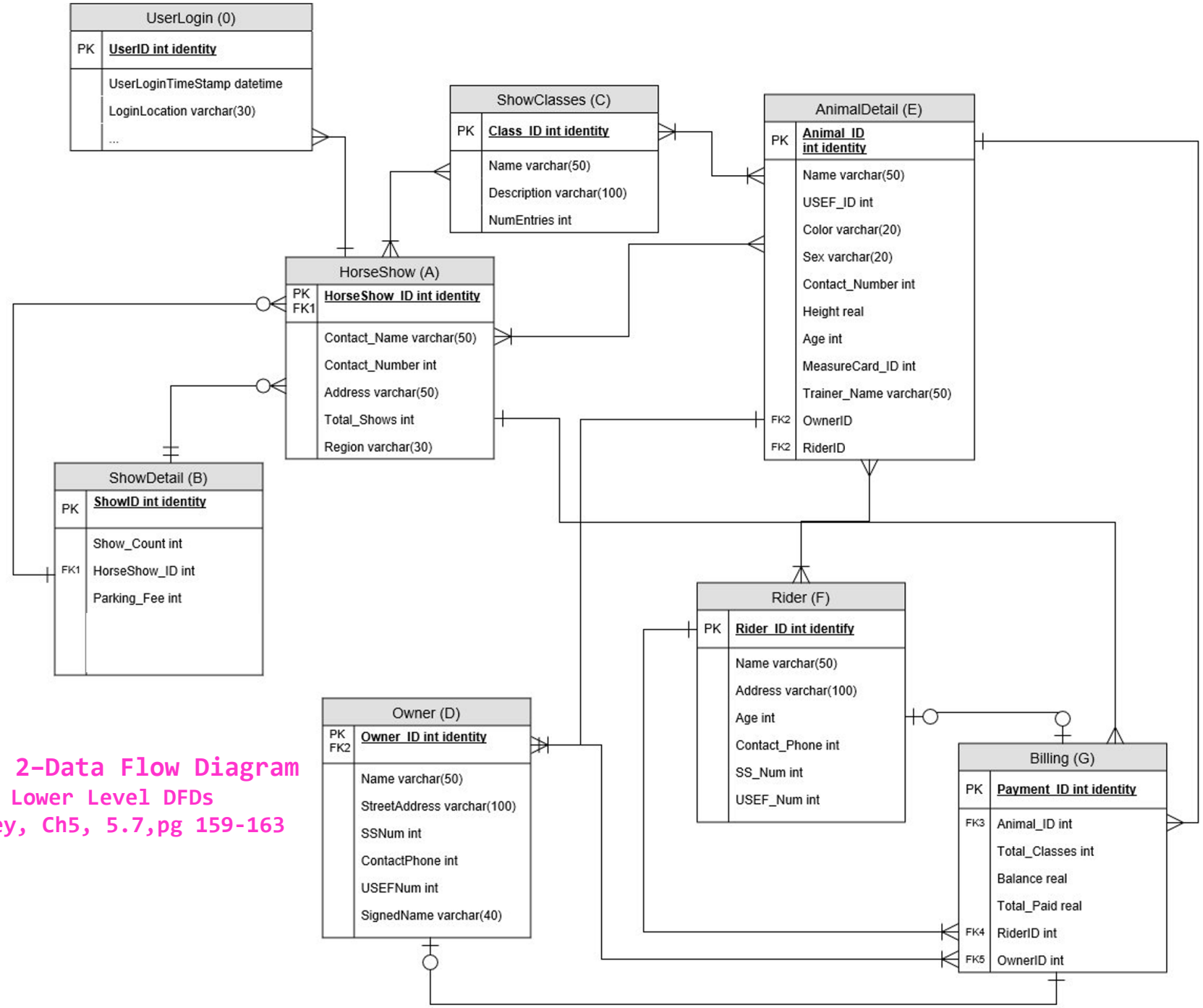
horseshow



Level 2-Data Flow Diagram

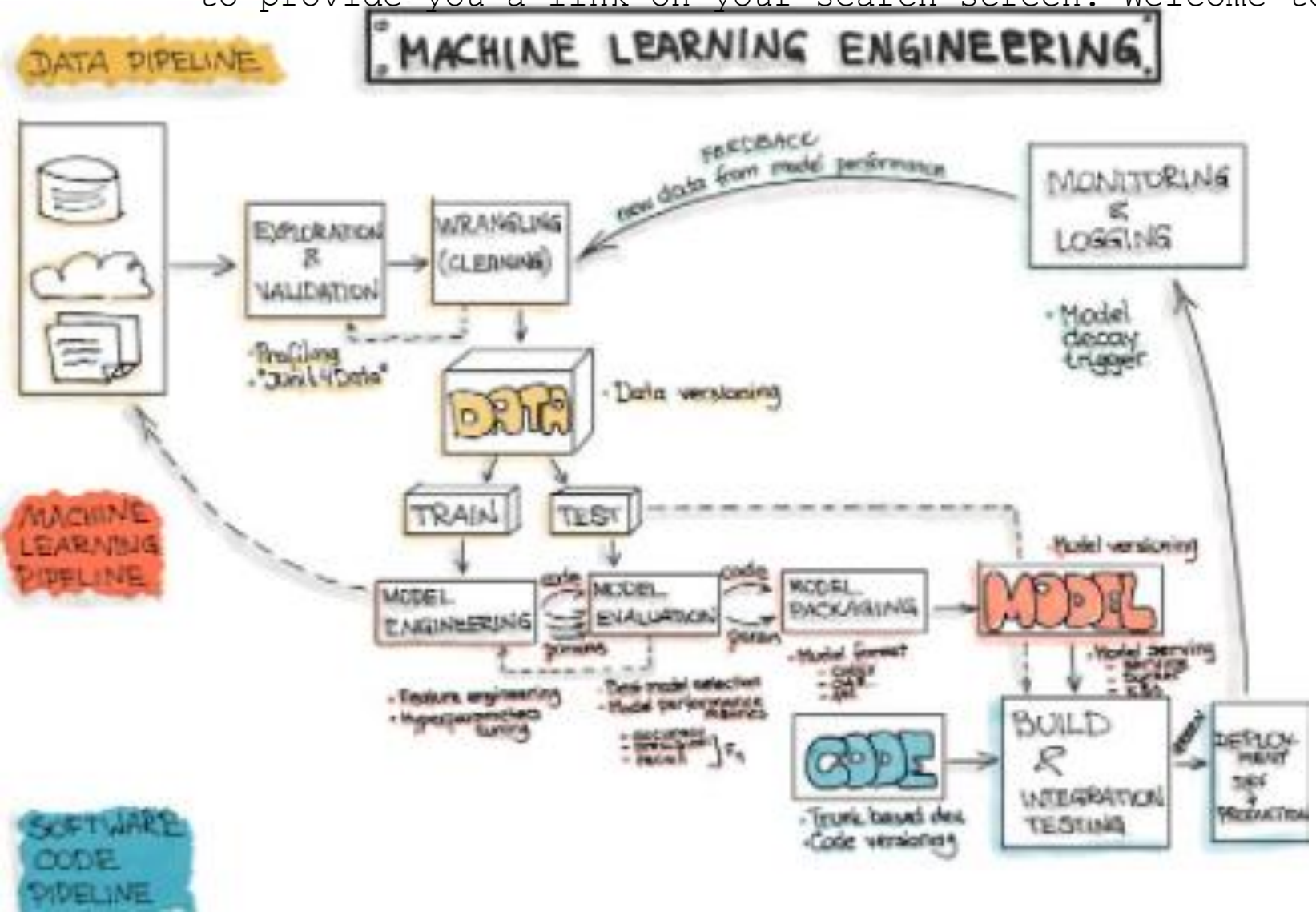
Lower Level DFDs

Tilley, Ch5, 5.7,pg 159-163



Model.6: Machine Learning Data Pipelines <website> <how.to.doc> <josh.Gordon.how.to.video> <wikipedia>

Purpose: data pipes execute in parallel or time-sliced fashion providing new clickbait or similar transaction behavior for an established ML model. New data kicks off further analysis as captured in the graphic. From a high level, if more people like you find a new Amazon "whizzy" somewhat of a "tizzy," then ML should indicate your profile as a potential buyer to provide you a link on your search screen. Welcome to modern business 101.



An Overview of the End-to-End Machine Learning Workflow

Generally, the goal of a machine learning project is to build a statistical model by using collected data and applying machine learning algorithms to them. Therefore, every ML-based software includes three main artifacts: Data, ML Model, and Code.

Corresponding to these artifacts, the typical machine learning workflow consists of three main phases:
1. Data Engineering: data acquisition & data preparation,
2. ML Model Engineering: ML model training & serving, and
3. Code Engineering :integrating ML model into the final product.

The Data Engineering pipeline includes a sequence of operations:
1. Data Ingestion
2. Exploration and Validation
3. Data Wrangling
4. Data Labeling
5. Data Splitting

Model Engineering
The Model Engineering pipeline includes a number of operations that lead to a final model:
1. Model Training
2. Model Evaluation
3. Model Testing
4. Model Packaging

Model Deployment
The final stage of the ML workflow is the integration of the previously engineered ML model into existing software. This stage includes the following operations:
1. Model Serving
2. Model Performance Monitoring
3. Model Performance Logging

from abhishek prasad but the linked in post page was torn

b.hogan@snhu.edu <bh.github> **Note:** Wikipedia is an information only reference. It is not an academic reference.

Model.7 Database Normalization:

<bh.github> <website> <how.to.doc>
<how.to.video> <wikipedia>

History:

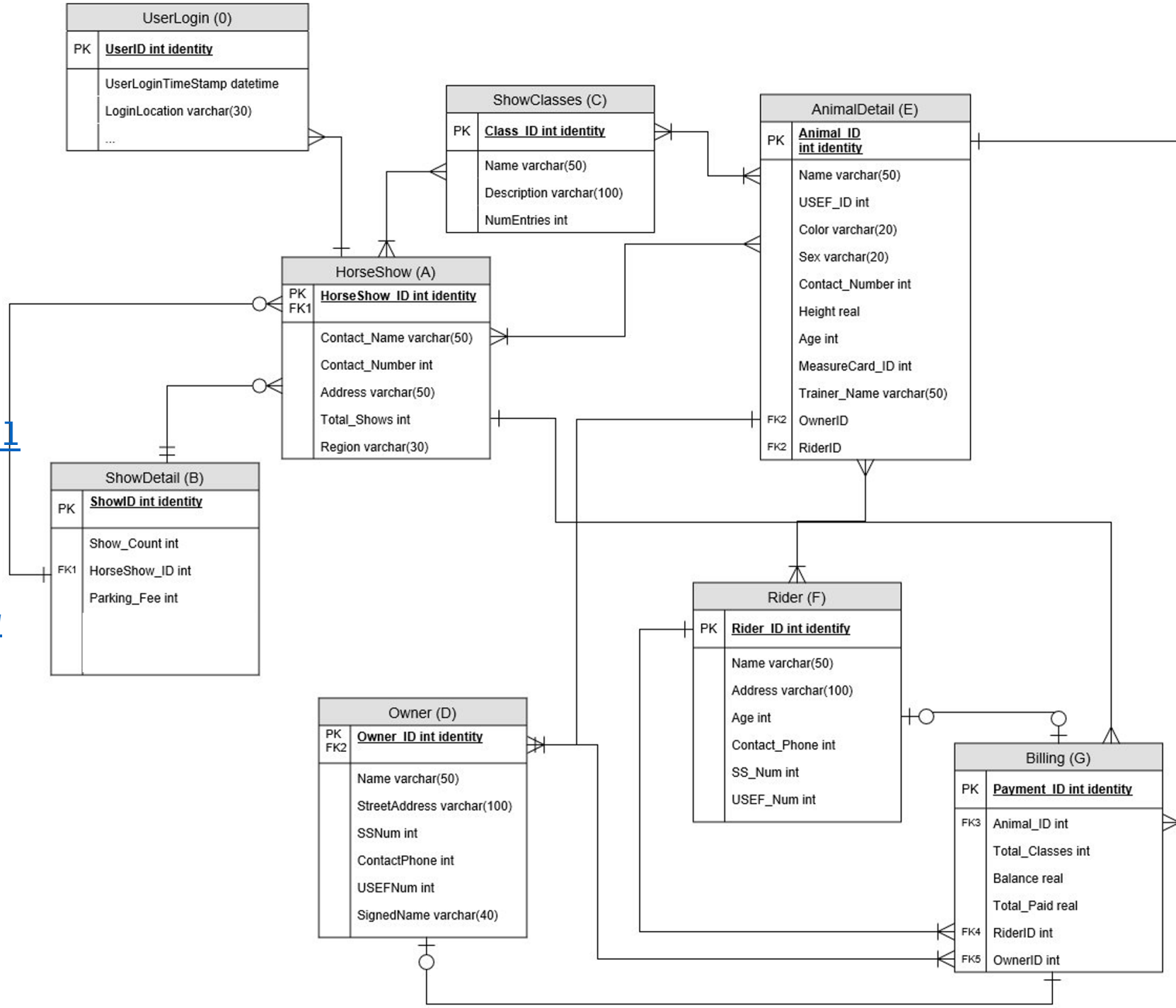
Purpose:

relational

database

tables

horseshow

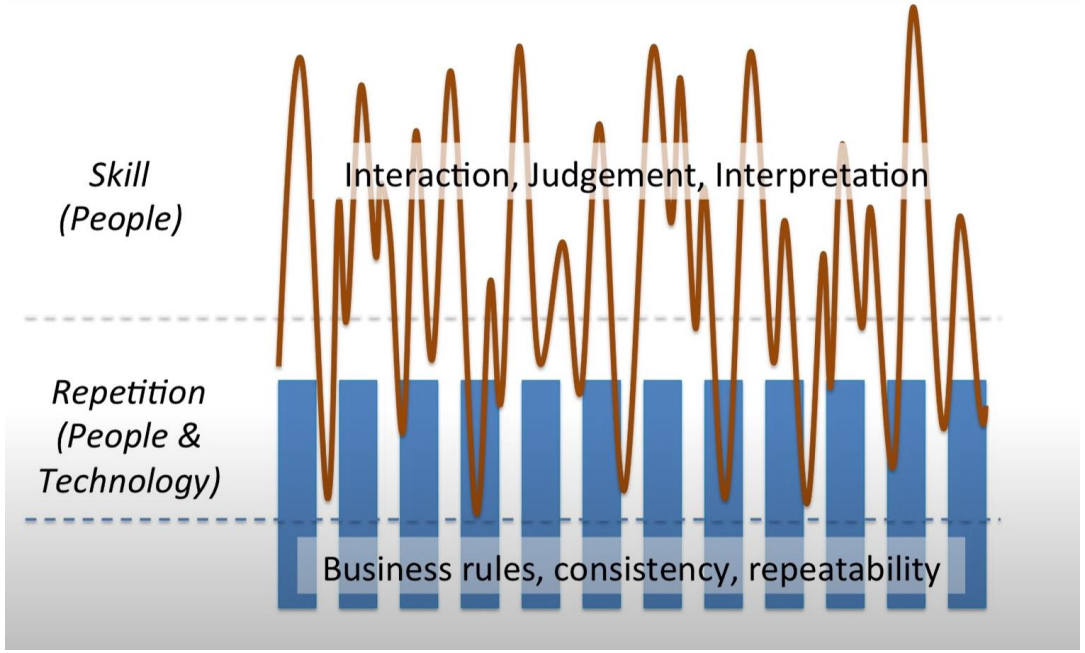


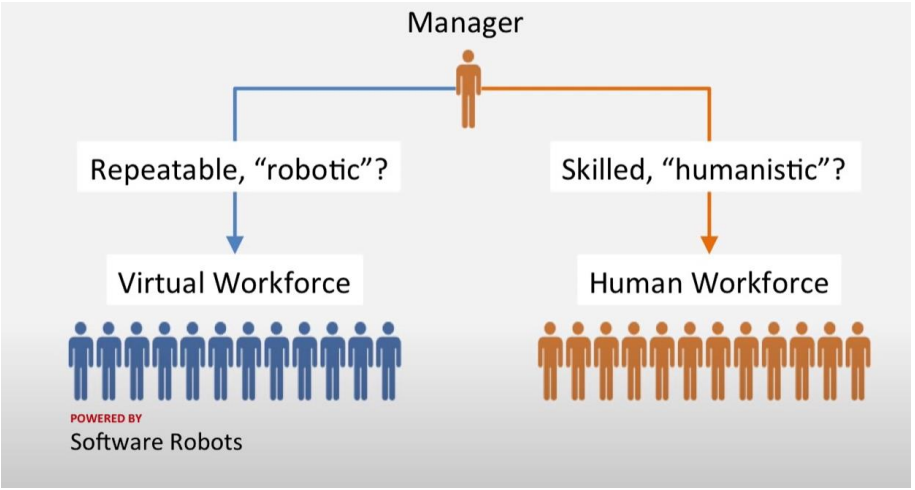
Model.8: Robotic Process Automation (RPA) <bh.github> <website> <IBM.vid.2022> <wikipedia>

History: Recent form of business process automation using software robots (bots) and artificial intelligent AI-digital workers to perform tasks user do by watching users perform the tasks in the GUI and then replicate it.

Purpose: Use of IT widgets to read emails, forms, to replicate human behavior and eliminate the "digital tape and glue" of cross updating amongst mix-matched current and legacy system applications. AI and machine learning can also help figure out and classify what the newly received information is and suggest where it should go instead of a human orchestrating it because systems have been codified, labeled, and connected to AI-digi-workers.

What are people doing in Reality?



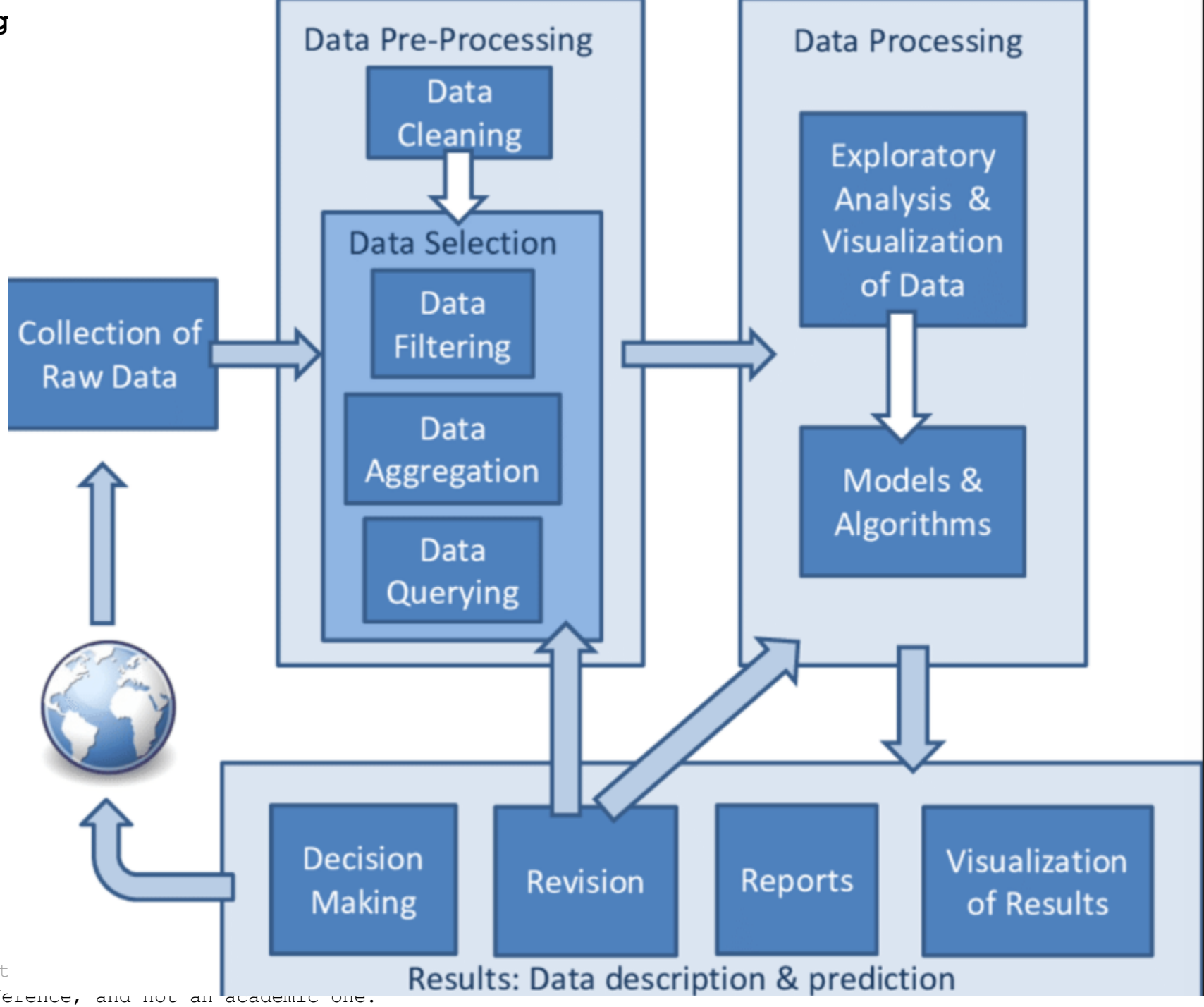


*Retrieved from: White Collar Robots: The Virtual Workforce | David Moss | TEDxUCL – YouTube, on 10.18.22.

Model.9: DataMart Decision Making

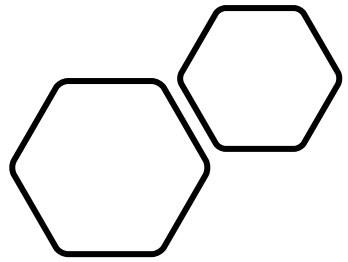
History:

Purpose:



b.hogan@snhu.edu. wikipedia is an information reference, and not an academic one.

subtrate



Python Skill.1:
Data Transformation

<bh.github>
<website>
<how.to.video>
<wikipedia>

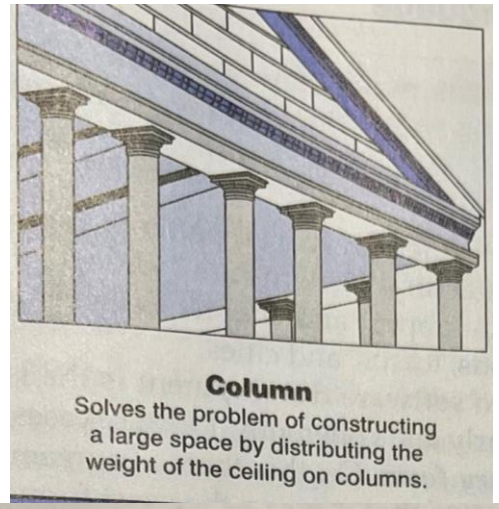
History: deep history and university dispute on ownership

Purpose: Apply

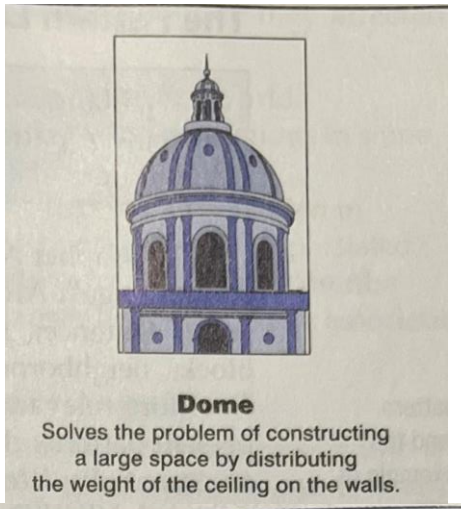
Model.10: Column vs Dome <bh.github> <website> <how.to.doc> <how.to.video> <wikipedia>

History: Use of classic pattern abstractions to form built to suit analysis approaches.

Purpose: abc



Column
Solves the problem of constructing a large space by distributing the weight of the ceiling on columns.



Dome
Solves the problem of constructing a large space by distributing the weight of the ceiling on the walls.

Patterns are old. A systematic way to describe patterns and their consequences is new.

Since time immemorial, people have adopted and adapted solutions found by other people. Figure 14.1 represents two architectural patterns that are so successful that they have dominated the building of large spaces for thousands of years. We can find numerous other examples for patterns in the history of human activities, from hunting to pottery and from shipbuilding to medicine. Such adaptations, however, were not usually called patterns (except when used as a synonym for "model," as in "dress pattern"), and no systematic way existed to describe them.

In 1977, Christopher Alexander, an architect, and his coauthors published *A Pattern Language: Towns, Buildings, Construction*. Rather than focusing on narrow technical or aesthetic considerations, the book explored hundreds of patterns that define the relationships among people and their architectural habitats: the themes that, for good or for bad, are repeatedly adopted by people or builders.

As it turned out, Alexander's greatest impact was not on architecture but on *software* architecture or, to be more precise, on object-oriented design. A systematic analysis and presentation of patterns requires distinct entities that only an object-oriented framework can provide.

Application of patterns to software design, both in theory and practice, started in the late 1970s and early 1980s. One of the first was the **Model-View-Controller** (MVC), an architectural pattern that separates the application into three logical components: data model, user interface, and control logic. (The general idea, if not the terminology, should be familiar to you by now.) The pattern, developed at Xerox's Palo Alto Research Center (PARC), was the foundation of the modern graphical user interface (GUI).

During the 1980s and early 1990s, many pioneers contributed to a growing awareness of patterns. One was Peter Coad, who in the article "Object-Oriented Patterns" offered certain guidelines and examples for identifying and adapting software patterns:

> *An object-oriented pattern is an abstraction of a doublet, triplet, or other small grouping of classes that is likely to be helpful again and again in object-oriented development.*
>
> Patterns are found by trial-and-error and by observation. By building many object-oriented models and by observing many applications of the lowest-level building blocks and the relationships established between them, one can find patterns.

What turned patterns into a serious subject in software development was the publication of *Design Patterns: Elements of Reusable Object-Oriented Software*, by Erich Gamma *et al* in 1997. (It has become known as the "Gang of Four" book, a reference to the number of its authors.) This book offered a viable framework for identifying and defining patterns that apply to software design.

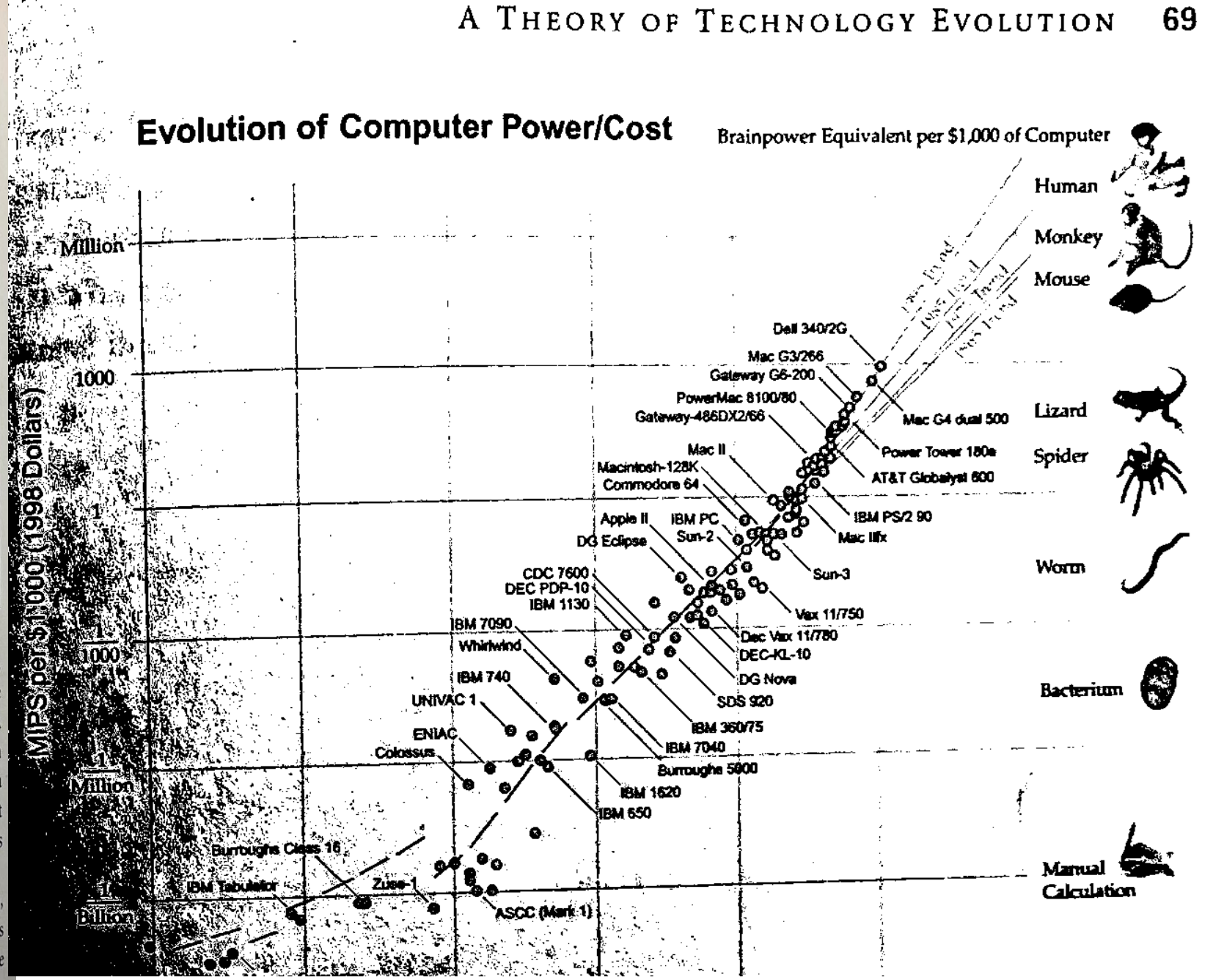
Since then the literature has grown constantly, with books that either expand on existing patterns or present new ones for new contexts such as analysis, enterprise architecture, modeling, etc.

Fractal Dimensions and the Brain

Note that the use of the third dimension in computing systems is not an either-or choice but a continuum between two and three dimensions. In terms of biological intelligence, the human cortex is actually rather flat, with only six thin layers that are elaborately folded, an architecture that greatly increases the surface area. This folding is one way to use the third dimension. In "fractal" systems (systems in which a drawing replacement or folding rule is iteratively applied), structures that are elaborately folded are considered to constitute a partial dimension. From that perspective, the convoluted surface of the human cortex represents a number of dimensions in between two and three. Other brain structures, such as the cerebellum, are three-dimensional but comprise a repeating structure that is essentially two-dimensional. It is likely that our future computational systems will also combine systems that are highly folded two-dimensional systems with fully three-dimensional structures.

Notice that the figure shows an exponential curve on a logarithmic scale, indicating two levels of exponential growth.[36] In other words, there is a gentle but unmistakable exponential growth in the *rate* of exponential growth. (A straight line on a logarithmic scale shows simple exponential growth; an upwardly curving line shows higher-than-simple exponential growth.) As you can see, it took three years to double the price-performance of computing at the beginning of the twentieth century and two years in the middle, and it takes about one year currently.[37]

Hans Moravec provides the following similar chart (see the figure below), which uses a different but overlapping set of historical computers and plots trend lines (slopes) at different points in time. As with the figure above, the slope increases with time, reflecting the second level of exponential growth.[38]



lecture_notes it304.wk7d1

lecture_notes
it304.wk7d1

Doubling (or Halving) Times[33]

Dynamic RAM "Half Pitch" Feature Size (smallest chip feature)	5.4 years
Dynamic RAM (bits per dollar)	1.5 years
Average Transistor Price	1.6 years
Microprocessor Cost-per-Transistor Cycle	1.1 years
Total Bits Shipped	1.1 years
Processor Performance in MIPS	1.8 years
Transistors in Intel Microprocessors	2.0 years
Microprocessor Clock Speed	3.0 years

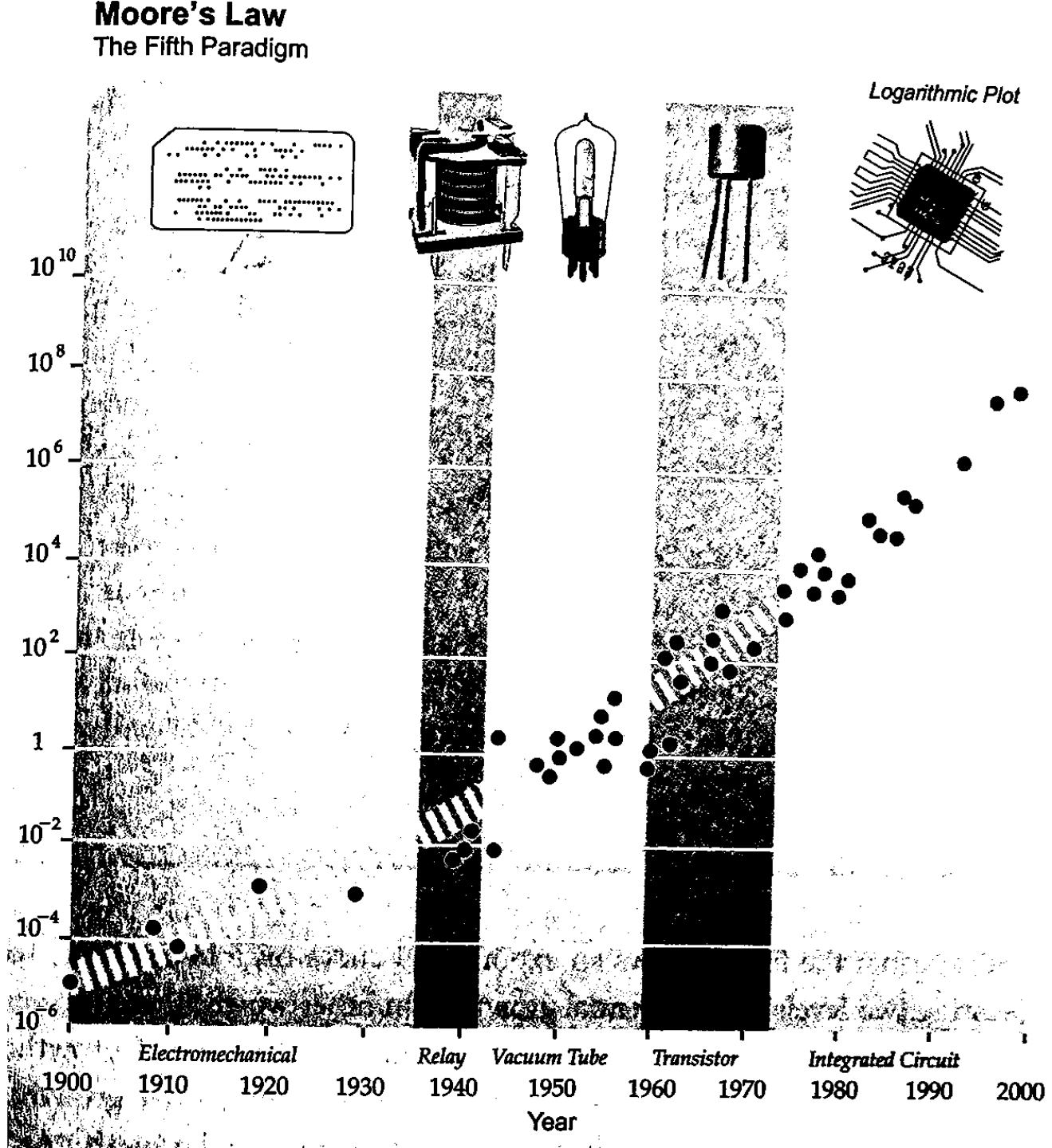
Moore's Law: Self-Fulfilling Prophecy?

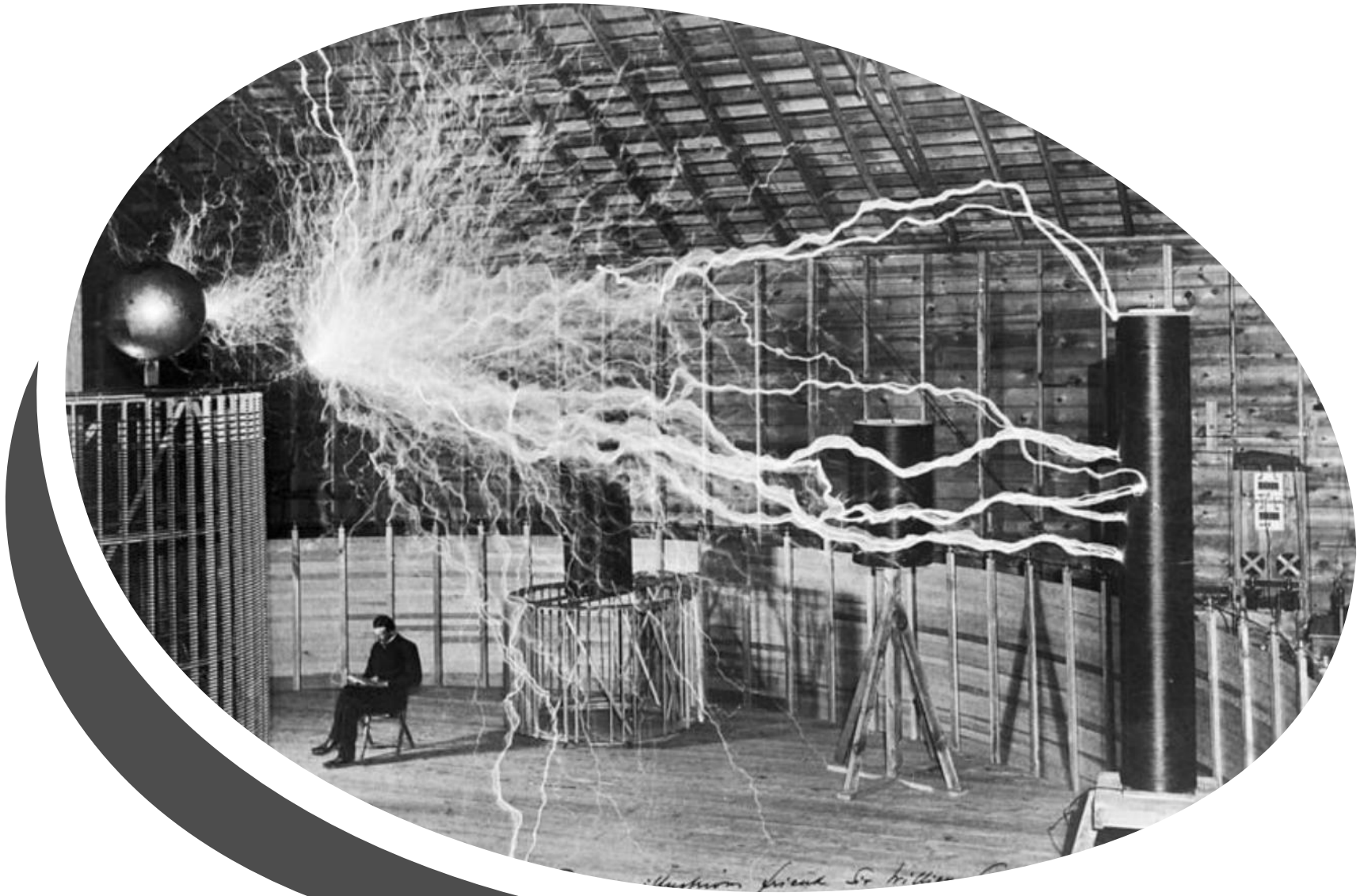
Some observers have stated that Moore's Law is nothing more than a self-fulfilling prophecy: that industry participants anticipate where they need to be at particular times in the future, and organize their research and development accordingly. The industry's own written road map is a good example of this.[34] However, the exponential trends in information technology are far broader than those covered by Moore's Law. We see the same types of trends in essentially every technology or measurement that deals with information. This includes many technologies in which a perception of accelerating price-performance does not exist or has not previously been articulated (see below). Even within computing itself, the growth in capability per unit cost is much broader than what Moore's Law alone would predict.

The Fifth Paradigm[35]

Moore's Law is actually not the first paradigm in computational systems. You can see this if you plot the price-performance—measured by instructions per second per thousand constant dollars—of forty-nine famous computational systems and computers spanning the twentieth century (see the figure below).

As the figure demonstrates, there were actually four different paradigms—electromechanical, relays, vacuum tubes, and discrete transistors—that showed exponential growth in the price-performance of computing long before integrated circuits were even invented. And Moore's paradigm won't be the last. When Moore's Law reaches the end of its S-curve, now expected before 2020, the exponential growth will continue with three-dimensional molecular computing, which will constitute the sixth paradigm.

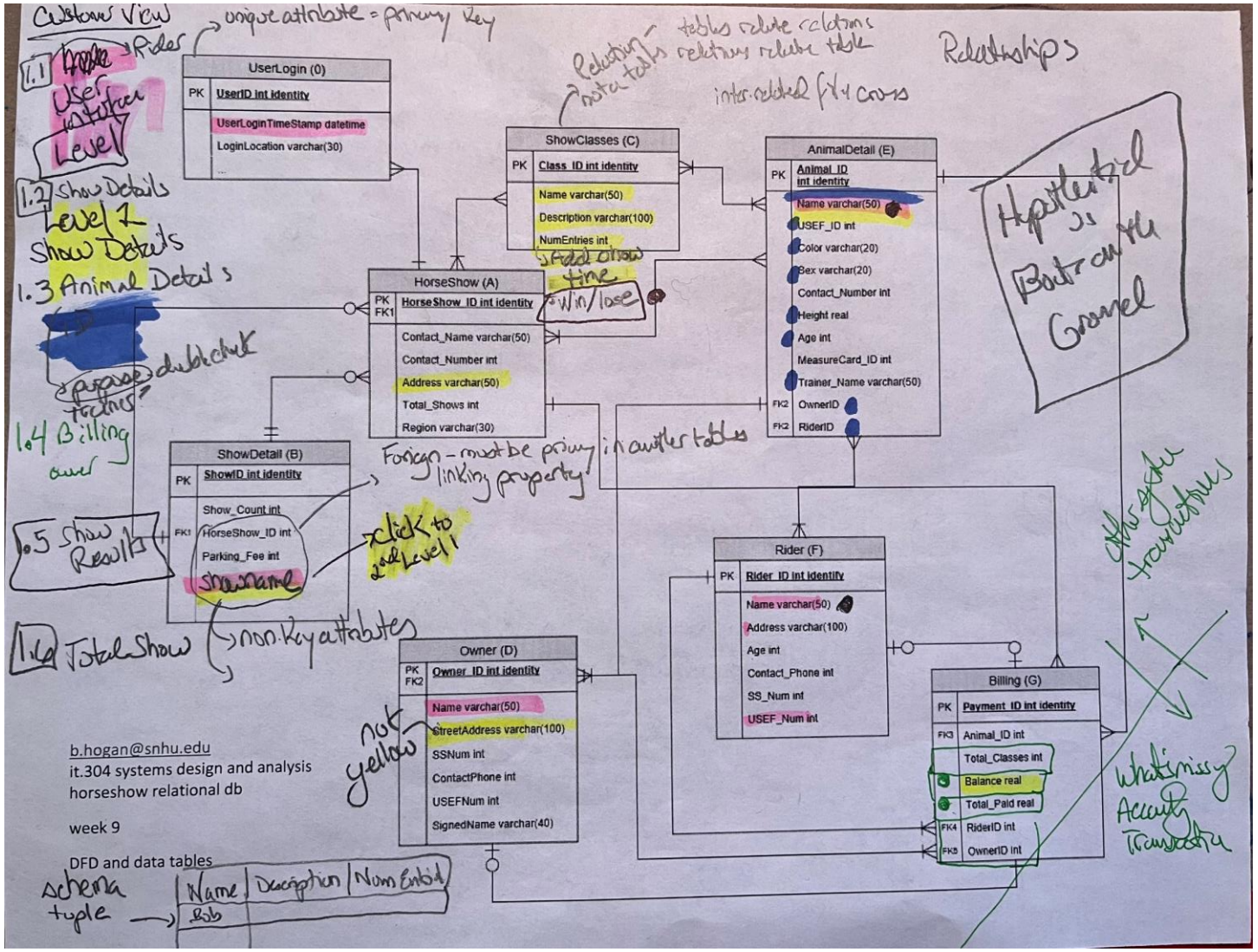




Model.5a: xx <bh.github> <website> <how.to.doc> <how.to.video> <wikipedia>

History: draft

Purpose:




```
3 or later of the GNU General Public License. GNU Screen is free software.
```

```
terminal multiplexer

ought of as a text version of graphical window managers, or as a way of putting virtual
login session. It is a wrapper that allows multiple text programs to run at the same time, and
that allow the user to use the programs within a single interface productively. This enables
tures: persistence, multiple windows, and session sharing.


iginally designed by Oliver Laumann and Carsten Bormann and published in 1987.[1]

ria included faithful VT100 emulation[dubious - discuss] (including ANSI X3.64 (ISO 6429) and ISO
easonable performance for heavy daily use when character-based terminals were still common. Later, the
e novel feature of disconnection/reattachment was added.

Oliver Laumann handed over maintenance of the code to Juergen Weigert and Michael Schroeder at the
ity of Erlangen-Nuremberg, who later moved the project to the GNU Project and added features such as
screen, cut-and-paste, and screen-sharing.[2]

See also

tal icon Free software portal
```

```
8   select 8
9   select 9

[Press Space for next page; Return to end.]
```

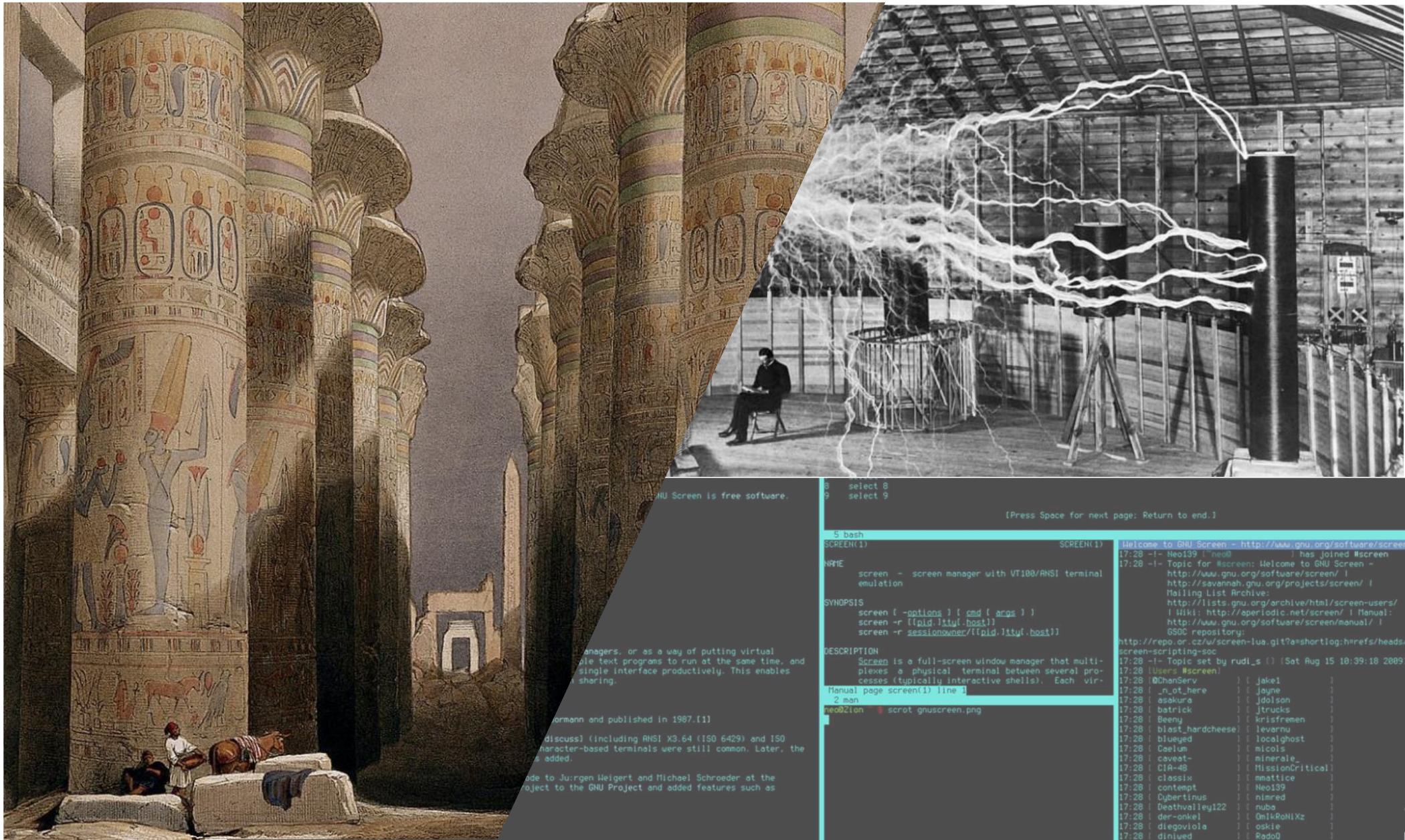
```
 5 bash
```

```
SCREEN(1)                                                       SCREEN(1)

NAME
       screen  -  screen manager with VT100/ANSI terminal
       emulation

SYNOPSIS
       screen [ -options ] [ cmd [ args ] ]
       screen -r [[pid.]tty[.host]]
       screen -r sessionowner/[[pid.]tty[.host]]

DESCRIPTION
       Screen is a full-screen window manager that multi-
       plexes  a  physical  terminal between several pro-
       cesses (typically interactive shells).  Each  vir-
Manual page screen(1) line 1
 2 man
neo@2ion ~ $ scrot gnuscreen.png
```

```
Welcome to GNU Screen - http://www.gnu.org/software/scree
17:28 -!- Neo139 [~neo0                ] has joined #screen
17:28 -!- Topic for #screen: Welcome to GNU Screen -
          http://www.gnu.org/software/screen/ |
          http://savannah.gnu.org/projects/screen/ |
          Mailing List Archive:
          http://lists.gnu.org/archive/html/screen-users/
          | Wiki: http://aperiodic.net/screen/ | Manual:
          http://www.gnu.org/software/screen/manual/ |
          GSOC repository:
http://repo.or.cz/w/screen-lua.git?a=shortlog;h=refs/heads/
screen-scripting-soc
17:28 -!- Topic set by rudi_s [] [Sat Aug 15 10:39:18 2009
17:28 [Users #screen]
17:28 [@ChanServ          ] [ jake1          ]
17:28 [ _n_ot_here        ] [ jayne          ]
17:28 [ asakura           ] [ jdolson        ]
17:28 [ batrick           ] [ jtrucks        ]
17:28 [ Beeny             ] [ krisfremen     ]
17:28 [ blast_hardcheese  ] [ levarnu        ]
17:28 [ blueyed           ] [ localghost     ]
17:28 [ Caelum            ] [ micols         ]
17:28 [ caveat-           ] [ minerale_      ]
17:28 [ CIA-48            ] [ MissionCritical]
17:28 [ classix           ] [ mmattice       ]
17:28 [ contempt          ] [ Neo139         ]
17:28 [ Cybertinus        ] [ nimred         ]
17:28 [ Deathvalley122    ] [ nuba           ]
17:28 [ der-onkel         ] [ OmIkRoNiXz     ]
17:28 [ diegoviola        ] [ oskie          ]
17:28 [ dinilved          ] [ RadoO          ]
```

Model.X: <bh.github> <website> <how.to.doc> <how.to.video><wikipedia>
History:
Purpose:

knowledge representation



bh.github
b.hogan@snhu.edu

GNU Screen split screen. At the left, links (web browser), at the right top, screen's help. At the bottom we have screen's man page, IRC client irssi, and a terminal taking this screenshot.

More details