TRIZ and Software - 40 Principle Analogies, a sequel.

Rob van den Tillaart, R&D
Oce Technologies B.V.
rvdt@oce.nl

Introduction

Some years ago there were two publications by Kevin C. Rea about the use of TRIZ in Computer Science and Information Technology called "TRIZ and Software - 40 Principle Analogies, Part I and II". See http://triz-journal.com/archives/2001/09/e/index.htm and http://triz-journal.com/archives/2001/11/e/index.htm

In these two articles there were a few gaps as there were no analogies and examples for some of the 40 inventive principles of Altshuller. In this short essay I will present additional software analogies and examples as a proposal to fill these gaps. I reused the layout used by Kevin Rea's in his articles as I found them useful and this way the new examples and analogies can be compared and merged more easily.

There are several actions still to do like base-lining definitions of the 40 principles for SW. As computer science is a very wide area the examples in this essay are from different areas, sometimes design level, sometimes on implementation level. It would be a good step to make separate 'TRIZ4'-overviews for the different parts of the software life cycle. Think of focussing on the requirement elaboration, analysis, architecture design, building, deploying and maintenance process. People can join this kind of discussions at http://trizforsoftware.com.

1 The 40 inventive principles

1.1 Segmentation

Dividing the system in smaller parts.

Software Analogy:
Divide a system, the design, an object or a data stream into autonomous parts. Think of modular design or the use of OO technology. Increase the level of granularity until a solution is reached. Bits can be thought of as atomic in the context of an encoding scheme.

Software Examples:

- Best known example is the top down approach of designing an application. At design time one divides the system into smaller chunks thereby dividing the complex system in smaller less complex modules. This process can be done again for every module until the building blocks aren't complex anymore.
- One can build an application upon autonomous services that are available either locally or over the Internet like Web-services. Similar one can build applications using commercial available components.

1.2 Extraction

Separate (extract) an interfering part or property from a technical system, or single out the only necessary part (or property).

Software Analogy:
Given a language, define a representation for its grammar along with an interpreter that uses this representation to extract/interpret sentences in the language.
Remove those parts of an information stream that add no value to it.

Software Examples:

- In signal processing there exist several ways to do noise reduction. For example one can calculate or measure the background noise in an image, subtract it from the image to enhance contrast of the real signal.
- The MP3 sound compression scheme removes those parts of the audio signal that most people cannot hear due to characteristics of the human hearing function.
- Lossy image compression e.g. JPEG removes information to get far better compression ratios of an image without (too) much image distortion.
- A tokenizer in a parser replaces every keyword with a smaller code to convert human readable code into code that can be processed by a machine. The simplest tokenizers are command line argument parsers that change a command line argument (a string) into a more elementary data type like an integer or even a boolean.

1.3 Local quality

Change a system from uniform to non-uniform; Make each part of a technical system fulfill a different and useful function.

Software Analogy:
Change an object's classification in a technical system from a homogenous hierarchy to a heterogeneous hierarchy.

Software Examples:

- In an AI application that uses a blackboard model, different workers try to solve (part of) a problem in different ways e.g. partly top down, partly bottom up.
- Replace a linked list of lion objects by a linked list of animal objects. Higher abstraction allows non-uniform objects to be stored in the same container class or vice versa.

1.4 Asymmetry

Change the shape from symmetrical -> asymmetrical

Software Analogy:
Bring asymmetry in an application. Change the asymmetry of a technical system in order to non-uniformly affect a desired result of a computation.

Software Examples:

- Let a program e.g. a filter be able to process different input formats without increasing the number of output formats or single format in, multiple out.
- Suppose one has to calculate the average of a long list. Normally one counts all the values and divides the sum by list size. If the answer does not need to be exact one can do the same for a randomly chosen subset. This approximation is often sufficient and much faster. (see also 2.16 partial or excessive action).
- Standard quicksort takes the middle element as pivot. By using the median of three one often gets better performance.
- Replacing binary search that samples the middle element of a sub-array with a weighted binary search by using statistical knowledge about the distribution of the elements. An example is a weighted binary search that takes into account the distribution characteristics of the characters of the alphabet when searching in some sort of dictionary.

1.5 Consolidation

Do multiple operations in parallel

Software Analogy: Make processes run in parallel.

Software Examples

- Multitasking operation systems, multithreaded applications, hyper-threading, multi processor applications.
- Seti@home is a massive parallel computation distributed over millions PC's in the world. These PC's ask for a piece of data to analyze and sent their results back to the server. The boinc.org project is a spin off of Seti@home that provides a framework for this kind of parallelisms.
- A packet switching network allows multiple data streams run side by side over one wire. In fact they are not really parallel but sequential interleaved.
- An optical network cable (glass fiber) can really have parallel communications as multiple waves of different frequency are sent through the fiber simultaneously.
- A web page can be constructed of multiple frames. Every frame can come from a different server. Fetching such a web page is much faster as multiple connections can be used in parallel.
- Printing multiple copies of a document can be done faster when one prints it in parallel on more than one machine.
- A raid-5 disk array that divides a file over multiple disks can retrieve and store the file faster as it is retrieved/stored in parallel.
- Some peer to peer file sharing applications see that the requested file is available at multiple remote PC's. Different chunks of the file can be downloaded from different PC's combining them locally. The effect is an increased throughput.

1.6 Universality

Make a part of object perform multiple functions

Software Analogy:
Make a technical system support multiple and dynamic classifications based on context.

Software Examples:
- There exist many (command line) tools that behave differently depending on their startup parameters. Look at the help of XCOPY for example under windows.
- The Norton commander embeds multiple different functions in one single application.
- A function that searches for the maximum value can easily be extended to search for the minimum as well. Especially when a lot of I/O is involved for reading the data from a file.

1.7 Nesting (Matrioshka)

Place objects into another

Software Analogy:
Inherit functionality of other objects by “nesting” their respective classes inside a base class.
Uses multiple layers of abstraction.
Use recursive data structures and algorithms.

Software Examples:
- Layered software architectures like the OSI model for network communications.
- The top down analysis & design approach: design the high level components first, thereafter the component becomes an ‘autonomous project’ with a defined interface.
- A matrix is an array of arrays. One can define multidimensional arrays.
- Recursive data structures: tree’s, linked lists, but also the good old directory with subdirectories.
- Recursive algorithms: quicksort, tree-walking algorithms, backtracking.

1.8 Counterweight

To counter the weight of a system, merge it with other objects that provide lift.

Software Analogy:
Use sharing to support large numbers of fine-grained objects efficiently to counter dynamic loads on a technical system.
Use hashing functions to distribute ‘weight evenly’

Software Examples:
- To improve the performance of web servers one can divide a page in multiple frames and store the frames on different disks or even different servers. One can distribute the (large) images or data-blobs over different disks instead of putting them on one disk. The performance will increase.
- With hashing functions one can distribute objects over a set of buckets while keeping the number of objects in a bucket roughly the same.

1.9 Prior Counteraction

Pre-load a counter tension to an object to compensate excessive and undesirable stress.

Software Analogy:
Perform preliminary processor actions in system that will improve a later computation.

Software Examples:

- Suppose a user must wait for a time consuming action. If one informs the user that the action takes 10 seconds and it takes longer stress levels will rise. If one says it costs 30 seconds and it takes less time the user will like it.
  Recall installation programs (or uninstallers) that say you only have to wait 2 seconds, but those 2 seconds can take 2 minutes sometimes or more.
- When an application starts that reads from a database one can load the most important tables or indexes ahead [can also be seen a 'Prior Action']

1.10 Prior Action

Perform, before necessary, a required change of an object (either fully or partially). Carry out all or part of the required action in advance.

Software Analogy: Same as above.

Software Examples;

- The read-ahead cache in a disk drive prepares for real reads.
- A web browser can fetch all references to other web pages that exist on the current page while the user reads the current page. A less memory consuming action is the resolving of the IP addresses of the hosts in the URL's on the page.
- Imagine a database application that prepares data sets before it is asked for as it recognizes patterns in the usage of the application.
- A printer driver that tests the availability of a printer before the users finishes his settings.

1.11 Cushion in advance

Prepare emergency means beforehand to compensate the relatively low reliability of an object.

Software Analogy:
Use an algorithm that handles worst-case harmful effects and maintains global invariance.
Don't assume anything about something you haven't completely in control. Whether it be 3rd party libraries, communication lines or users providing data.

Software Examples:

- An operating system that makes backups of essential files before it can be used. This prevents reinstalling when something gets corrupted.
- Set a timeout upon an operation that could block the system. E.g. a query to a database or trying to connect to a network service.
- The use of transactions in databases that can be rolled back are also a sort of cushion.
- An input handler that recognizes and filters or ignores badly formatted input.
- A route planner that calculates alternative routes during a ride. It knows from history that one or more parts of the route are often blocked by traffic jam. This way alternative routes are instantaneous available when needed.
- A word-processor that makes intermediate snapshots of the document typing to prevent loss of data in case of an OS crash. This includes making a copy of the document before editing.
- The trash can on most modern computer desktops

1.12 Equipotentiality

In a potential field limit position changes

Software Analogy:
Change the operational conditions of an algorithm so as to control the flow of data into and out of a process.
An analogy of the potential field is the time it costs to recreate an object in memory. Objects that costs much time must be kept alive as long as possible.

Software Examples:

- Intelligent memory swapping algorithm that calculates the chance of potential usage of an object in main memory. It will swap those objects that have the lowest potential usage.
- Keep intermediate results that took a long time to calculate as long as they are valid.
- In general it can be used when applications use data in a non-deterministic way. Try to keep the elements that most likely are used again.
- In genetic algorithms one often works with a 'gene' pool of possible solutions. Genetic algorithms keep the most potent solutions to 'breed' with. Solutions with less potential are removed.
- Reorder a database or file only when explicit needed. Often data is marked as deleted by flagging it as deleted. The records in the database aren't moved although they seem to have new positions in the database.

1.13 Do it in reverse

Invert actions to solve the problem.

Software Analogy:
Store transactions in reverse order for backing out.

Software Examples:

- Every scheduling algorithm does this as reversing a schedule can be faster.
- In some algorithms one works with multiple sets of intermediate results. Do the most strong restrictions on these sets as soon as possible, thereby keeping the intermediate sets small and this processing fast. e.g. when joining two tables in a database first restrict each table to a minimal set before calculating the Cartesian product,
- Some calculations can cause an Overflow or underflow. By changing the order of the calculations this can be prevented. A simple floating point formula p = x * y / z; can be written as p = (x / z) * y or as p = y/z * x; One can even choose between formulas depending on the relative values of the variables x, y and z;
- For poems one reverse the strings in a dictionary. This makes it easier to look for words that end the same way.

1.14 Spheroidality

Replace linear parts with curved parts, flat surfaces with spherical surfaces, and cube shapes with ball shapes.

Software Analogy:
Replace linear data types with circular abstract data types. Also consider tree based or graph based data types. Similar is valid for the algorithms: replace linear performing algorithms with non-linear ones.
In general replace the simplest solution with a more complex one that might have additional benefits.

Software Examples:

- Replace a linear search, with a more complex search like binary search.
- Replace binary search with 'weighted' binary search.
- Change coordinates from Cartesian to Polar format to ease some calculations.

- Replace printing from PC to printer with printing through a server which gives more control over the print flows. Not using the shortest path has extra benefits.

1.15 Dynamicity

Allow or design the characteristics of an object, external environment, or process to change to be optimal or to find an optimal operating condition..

Software Analogy: Same as above.

Software Examples:

- Imagine the intelligent cache that tunes its size based upon the usage of the last 10 minutes.
- A dynamic user interface of an application can hide less used functionality and may even focus on the next anticipated step.
- A mathematical parser which is able to reduce a complex formula before calculating it e.g. $y = \sin(x)/\cos(x) \Rightarrow y = \tan(x)$.
- When doing image processing, one can dynamically change the threshold locally to obtain optimal contrast for character recognition.
- A compiler reorganizes the parsing tree to optimize the code it generates.
- Change the view in an application like PowerPoint. Editing the text in the outline mode is easier and gives better overview than in the presentation mode.
- Some graphic producing programs let the graphics self adjust their scales.

1.16 Partial or Excessive action

When 100% is difficult to reach, slightly less or more may be sufficient.

Software Analogy:
Increasing the performance of measurable and deterministic computations by perturbation analysis.

Software Examples:

- Heuristic optimizers for NP complete problems like the travelling salesman that stop when they are at 95% of the optimal route. To get the last 5% would cost an non-equivalent amount of time.
- To sort large arrays often quicksort is used as this algorithm performs generally well. However when the sub-arrays to be sorted become small, say less than 10 elements, other sort algorithms are used as they perform better.
- To improve reliability of some spacecraft multiple different processors calculate the same value with different algorithms. If the results are the same chances are high that the result is right.

1.17 Transition into a new dimension

Move an object into another dimension or add an extra dimension.

Software Analogy:
Use a multi-layered assembly of class objects instead of a single layer.

Software Examples:

- A queue that grows and cannot be hold in memory due to size can put new elements on disk until there is room in main memory again.
- In a user interface with essential 2D windows add a 3rd dimension to place less used screen further in the background. Let the windows rotate in the 3rd dimension.
- In image processing change a black and white image to a grayscale image before reducing or enlarging the image.
- Use hexadecimal notation for values can make bit operations more understandable.

1.18 Vibration

Use oscillation.

Software Analogy:
Change the rate of an algorithm execution in the context of time until the desired outcome is achieved.
Another ‘computer synonym’ for vibration is alternating.

Software Examples:

- In a pipeline architecture in which several filters process data after another the amount of time scheduled for each filter is related to the processing time of an item and the length of a queue.
- In the same pipeline architecture one can schedule the processes in the same order as the pipeline to get some sort of "wave" through the pipeline. In this ‘resonance’ mode the data for the next filter is available in memory. Performance will be far better than with random ad hoc scheduling.
- Connect the refresh of a screen not to the ticking of the system clock but to the arrival or departure of data in the GUI.
- The alternating use of MIN-MAX algorithms in games and in artificial intelligence is also some sort of vibration.
- Another alternating effect is a processing algorithm that reads alternating from multiple queues, e.g. in a round robin way.
- To minimize average seek time on disks one let the read head oscillate form the first rack to the last and back again, serving all requests for sectors when the read head flies over them.
- New ‘disk’ technology that uses two vibrating squares, one full of read heads, in stead of rotating disks with one moving read head.

1.19 Periodic action

Use periodic or pulsating actions instead of continuous action.

Software Analogy:
Instead of performing a task continuously, determine the time boundaries and perform that task periodically.

Software Examples:

- Update a user table in a database just once a day and recalculate the sorting index just once per day instead for every individual update. Similarly remove entries flagged deleted only once per day.
- Prepare a mail merge on disk before sending it to the printer. Especially when "calculating" one page takes longer than printing it.
- Garbage collectors that only execute when the ‘garbage’ is above a certain threshold.

1.20 Continuity of useful actions

Try to perform 100% all the time, eliminate idle time

Software Analogy:
Develop a fine-grained solution that utilizes the processor at full load.

Software Examples:

- Multitasking operating system. All kind of processes can run in the background, like running queries, printing, disk de-fragmentation, virus scanning etc
- Seti@home screensaver makes usage of idle processing of the PC.
- During text editing one can do grammar and spelling check simultaneously.
- Auto completion of words during typing, propose parameters for formula's in a spreadsheet.

- Continuous compilation of the code in a programmer's editor.
- Continuous checking of the validity of URL's in your browser bookmarks,
- Checking validity of email addresses in your email application.
- In a flat bed scanner the scanner light moves twice over an image. Use the first scan to do a pre-scan to prepare optimally for the final scan (e.g. background correction). Or one can 'add' the two images to get a higher resolution.
- Decompress ahead when reading a compressed stream.

1.21 Rushing through

Conduct certain process steps at high speed.

Software Analogy:
Conduct the transfer of data in a burst mode just before a worst-case scenario.

Software Examples:
- Try to rewrite loops in software in such way that they fit in the cache of the main processor. This way the processor does not need to wait for memory I/O.
- Disable access to a database when doing a backup. The backup can be done faster and after the backup people use the database.
- When detecting a power outage do a memory dump to disk to save all data.
- When the frequency of disk errors rise make a full copy of a disk to another one a.s.a.p.

1.22 Convert harm into benefit

Eliminating a harmful action by adding another harmful action.

Software Analogy: Inverse the role of the harmful process and redirect it back.

Software Examples:
- If the software knows some calculation or query takes a long time, display a message that the user might fetch coffee or doing stretch exercises. This may even take more time but the user is not in an active waiting state.
- Take multiple sensor readings (= more work) and take the average to cancel noise (= better quality).
- If a program is just a bit too slow to be interactive, make it a batch processing program so the expectations of the user are quite different.
- If a program detects a bad quality communications line lower the communication speed. Although slower the final result can be a higher throughput due to less errors.
- If the quality of a storage device is low add extra bytes to the data in such a way that the data becomes error correcting.

1.23 Feedback

Introduce feedback into the system

Software Analogy:
Introduce a feedback variable in a closed loop to improve subsequent iterations based on qualifiers.

Software Examples:
- When some calculation or query takes a lot of time display a progress bar.
- A route planner can display the first results while it tries to optimize the route.
- An approximation algorithm can calculate something to a certain precision and intermediate values are presented on the screen.
- Genetic algorithms can show the best results found so far while iterating over the generations of 'solutions'.

1.24 Mediator

Use an intermediate carrier or process

Software Analogy:
Use a mediator to provide a view(s) of data to a process in the context of the processes application space.

Software Examples:

- The mediator design pattern does this explicitly. Instead of knowing the state of several objects and keep them up to date with my changing state I only need to inform the mediator or ask the mediator. This reduces communication between objects and reduces complexity. E.g. create a separate object for handling all configuration issues.
- Let objects in memory not directly communicate with database tables directly but let them access through a mediator so if the database changes this change does not ripple though the whole software.
- Use of a print server to hold jobs temporary when the printer is offline.

1.25 Self service

Make an object serve itself

Software Analogy: Same as above.

Software Examples:

- A security or secure application e.g. firewall or virus scanner that checks its own integrity before running.
- A database that is able to repair its own contents as it holds transactional information.
- A web browser that can redirect you too another web site when the requested server is not available. E.g. if Google is off line it reverts you to Yahoo.
- A web browser that proposes logical permutations/derivations (spell checked) variants of the hostname and shows these as alternatives if the hostname is not found.
- A clock object that synchronizes itself by means of the NTP protocol.
- An operating system that increases its own swap space.
- A backup program that includes the restore program in the backup; e.g. self extracting zip-files.
- Computer viruses or worms that can copy themselves over the internet

1.26 Copying

Use simpler inexpensive objects instead of expensive ones.

Software Analogy:
Instead of creating a new object that takes unnecessary resources perform a shallow copy.

Software Examples:

- Instead of building a new application from scratch (=expensive) reuse existing components (=cheaper). A pipeline of (Unix) commands is the classic example.

1.27 Dispose inexpensive short living objects

Replace an expensive object with multiple inexpensive objects, compromising certain qualities.

Software Analogy: Same as above.

Software Examples:

- Instead of using lot of high speed memory use disk space. Storage costs are exchanged with access time.
- Present a complex page not with all details as a completely correct preview takes a lot more time to calculate.

- Use low resolution for images when possible.
- Calculate with integers instead of floating point numbers. It is much faster and may be exact enough for your application.
- Replace an expensive function with a lookup table, e.g. replace sin(x) and cos (x) with a table that maps the calls on an array [0..45].
- Build a supercomputer from multiple old PC's.

1.28 Replacement of a mechanical system

Replace a mechanical means with a sensory.

Software Analogy: Same as above?
Software sec has no mechanical means. But for systems containing both mechanical an software components there are a lot of examples.

Software Examples:
- A doctors stethoscoop can be replaced by a microphone and an headphone. The advantage is that a recording can be made, complex signal processing can be done, e.g frequency analysis, heartbeat counting etc.

1.29 Pneumatic or Hydraulic construction **)

Use gas and liquid parts instead of solid parts

Software analogy:
Use more flexible or generic data-types instead of fixed data-types or even constants.

Software Examples:
- Uses variables instead of hard coded constants. Initialize these variables at start up with the appropriate value. When the constant changes (how could they) the system need only a change in one place.
- Use a dynamic array or a linked list instead of an array with fixed dimensions. Far less growing pains, or buffer overflow errors.

1.30 Flexile films or thin membranes

Isolate object from the environment

Software Analogy: Same as above.

Software Examples:
- OO Classes with defined (fixed) public interface can change their inner representation.
- The proxy pattern acts as a shield for one or more objects.

1.31 Porous materials **)

Make an object porous or add porous elements
[Increase reaction speed by increasing surface; or give access to internals]

Software Analogy:
Change the interface of a class exposing more or less of its internal structure.

Software Examples:
- An object which shows more of its inner workings, e.g. one can see and manipulate the internals of an object directly. Although it can corrupt integrity of the object it can increases performance to needed levels. E.g. one can store values in an object bypassing the validation of these values.

- By exposing more details of a storage medium an application can make assumptions about behavior. Storing data on a network disk is slower than a local disk, Unless block size of writes is optimized.

1.32 Changing the color

Change the transparency or color of an object or its external environment

Software Analogy: Same as above

Software Examples:
- A 90% transparent clock in the desktop that becomes less transparent when it approaches the time of an appointment.
- A monitoring application that is transparent on the GUI of the PC becomes less transparent when attention is needed.
- There exist applications for astronomers that can change the colors of the user interface to black and red instead of full color. This is needed when used outside in the dark as the human eyes needs time to get used to the dark to see the stars. A red GUI doesn't interfere with the sensitivity of the eyes. Could be the ultimate battery saver for laptops, work in the dark with no back-light with only red characters ☺.

1.33 Homogeneity

Make objects interact with a given object of the same material.

Software Analogy:
Create classes not as islands but in class hierarchies as this unifies different classes to look the same on a certain level. At that level they share the same properties.

Software Examples:
- Write algorithms that accept base-class objects as parameters so all derived classes can be processed too.

1.34 Rejecting and regenerating parts.

Discards portions of an object that have fulfilled their functions or modify these directly during operation. Conversely, restore consumable parts of an object directly in operation.

Software Analogy: Discard unused memory of an application.

Software Examples:
- Keep a pool of unused (complex) objects that are frequently created, used and deleted. By keeping a pool of them the 'creation' process is much faster as the objects are fetched from the pool. It also reduces the strain on the garbage collector for deleted objects.

1.35 Transformation properties.

Change degree of flexibility.

Software Analogy: Same as above.

Software Examples:
- Merge software applications that have almost identical databases to reduce the amount of redundancy.
- By working with templates an editor can become multipurpose, e.g. loading another syntax highlighter template makes an editor suitable for the other programming language.

1.36 Phase Transition **)

Use phenomena that occur during phase transitions,

Software Analogy:
Use intermediate results and objects that are available during a state transition of an object.

Software Examples:
- The changing of the state of an object might take quite some complex calculations or need expensive lookups in a database.
- By assigning intermediate results to some variables (caching) these intermediate results are available for other objects thereby possibly improving performance. E.g. a linked list can keep a pointer to the last element inserted. When one needs to search through that linked list one can check this pointer if it is wise to start at this node in the list or one has to restart from the begin of the list. This speeds up (lineair) searching with a factor 2 on average.

1.37 Thermal Expansion **)

Use thermal expansion or contraction of materials

Software Analogy:
Use data compression decompression to increase decrease size of data. Can also be seen as change of precision

Software Examples:
- When data is compressed (thermal contracted) it can be send faster over a network, or it takes less storage space. One has to heat it up (decompress) to make it usefull again.
- When one has to calculate with dollars, or other currency this is often a floating point calculation. By multiplying all numbers with 100 (heat them with a factor 100) one can calculate in the integer domain which is faster and gives no (less) rounding errors.
- Make buffers and caches dynamically (self?) sizeable so they can contain more or less information (more less energy) and make them optimally fit the available memory.

1.38 Accelerated oxidation

Replace common air with oxygen enriched air,

Software Analogy:
Use optimized representation for faster processing.

Software Examples:
- Replace an algorithm written in a high level programming language with a expert tuned, hand optimized, processor specific assembly routine. Or even with hardware that performs the same function. E.g. video compression.
- Write data on disk in the same order that the processor expects it, e.g. high endian versus little endian integers.
- Use binary formats to increase efficiency for processing, storage and network traffic.

1.39 Inert environment **)

Replace a normal environment with an inert one.

Software Analogy:
Replace a normal computing environment with a more restricted one.
[Sand-boxing, honey-potting]

Software Examples:

- Test software in a controlled environment so that if things go wrong the damage is restricted to the controlled environment.
- Install a dummy 'not protected' PC on the Internet to detect the early out brake of a computer virus or worm. [honey potting]
- Use simulation or simulated classes for some objects, e.g. simulate a file system to see the effects of an application.
- Test a new computer virus in an isolated PC to see what it does for harm.
- Restriction of functionality of executable components in a web browser.
- Run an application in a debugger in a step mode
  [not completely inert but far more controlled]

1.40 Composite materials

Change from uniform to composite (multiple) materials.

Software Analogy:
Change from uniform software abstractions to composite ones.

Software Examples

- <none>

**) In the original article of Kevin C. Rea there was no analogy for this inventive principle.