StrEmbed-5-6 Users' Manual

Embedding design structures in engineering information

Hugh Rice ([h.p.rice@leeds.ac.uk](mailto:h.p.rice@leeds.ac.uk)) and Hau Hing Chau ([h.h.chau@leeds.ac.uk](mailto:h.h.chau@leeds.ac.uk))

University of Leeds

July 2021

What is StrEmbed-5-6?

Structure Embedding version 5-6 (StrEmbed-5-6) is a deliverable from the Design Configuration Spaces (DCS) project, funded by the Engineering and Physical Sciences Research Council (grant reference EP/N005694/1).

Engineers use design structures, such as bills of materials (BoMs), to tailor product definitions, including shape, for particular activities. For example, an engineering BoM defines the as-designed product whereas a manufacturing BoM defines the as-built state of the same product and a service BoM includes information on how the product has been maintained. All of these BoMs relate to the same designed product. However, in practice, because of restrictions arising from current computer-aided design technologies and associated business systems, different BoMs are usually related to separate digital definitions of the same product. This creates significant data management problems that add cost, time and rework into product development processes. If resolved, substantial business benefits, through improved efficiency and effectiveness of product development processes, could be achieved.

StrEmbed-5-6 allows assembly/BoM structures to be (a) loaded from STEP or STP files and (b) manipulated via a number of operations: aggregate, disaggregate, remove part, add part, flatten, collapse and shuffle/sort/move parts or sub-assemblies. The assembly structure is visualised as a tree/directory, a lattice/network and as images (for part selection) and a 3D CAD-style viewer. This functionality allows the user to create unlimited new BoMs from (a) existing ones or (b) from scratch.

The software represents a step towards a full assembly creation and comparison tool that will allow a design structure to be embedded onto a corresponding order lattice. An order lattice is representation of a partially ordered set (poset) of the constituent parts of a product – the partial ordering being part-whole relationships between parts and sub-assemblies - which is equivalent to a Boolean algebra. This algebra has a number of useful mathematical properties that allow generation of alternative design structures for each stage of product life cycle, *e.g.* “as designed”, “as manufactured” and “as maintained”. This embedding relationship can be visualised using a Hasse diagram, and any combination or permutation of an alternative design structure can embedded in the same lattice. The final version of the software will allow: (a) design structure to be exported to a common data format, for use in other CAD systems; and (b) reconciliation of multiple BoMs of the same product that have diverged during the product development process.

How to install and run StrEmbed-5-6

Software details and operating system dependence

StrEmbed-5-6 is written in Python 3.8 and was developed in the [Spyder IDE](https://www.spyder-ide.org/) and, as such, can be run on any operating system. However, it has only been tested on Microsoft Windows 10 64-bit, and the executables provided (see below) are Windows-specific. Please contact the developers if you encounter problems when running StrEmbed-5-6 on an operating system other than Windows 10 64-bit.

If you use another operating system or a version of Python 2.x and have difficulty running StrEmbed-5-6, please contact the authors. StrEmbed-5-6 relies on a large number of packages (dependencies), the principal of which being: the cross-platform GUI interface module [Wxpython](https://www.wxpython.org/) that relies on or mimics the native controls of the user’s operating system; and a 3D viewer based on the [Open Cascade](https://www.opencascade.com/) framework *via* the module [PythonOCC](https://github.com/tpaviot/pythonocc).

Running StrEmbed-5-6 as an executable (recommended) and creating your own

Standalone executable versions of the software can be found at the [Github repository](https://github.com/paddy-r/StrEmbed-5-6), *via* the “Latest release” tab, and can be run directly and do not require any additional installations. The executables were created using the Pyinstaller module, and are OS-specific. You can create an executable for your particular OS by running Pyinstaller within your Python IDE, using the source files provided at the Github repository, *via* the process below.

Build your own StrEmbed-5-6 executable

You can build an executable from source using [Pyinstaller](https://www.pyinstaller.org/), as follows, in the command line:

pyinstaller --onefile --noconsole StrEmbed\_5\_6.py

where the --onefile tag creates a single executable file (recommended) and the --noconsole tag (optional) creates an application without a console window. Pyinstaller should be rerun and the latter tag removed if any errors occur during file creation, to allow debugging. More advice on installing and running Pyinstaller is available [here](https://pyinstaller.readthedocs.io/en/stable/operating-mode.html).

Running StrEmbed-5-6 as a Python script in a Python IDE

Download or clone the repository from Github [here](https://github.com/paddy-r/StrEmbed-5-6), ensuring the “Images” folder is at the same path as the main script, StrEmbed\_5\_6.py, and the secondary modules, step\_parse\_5\_6.py.

Using StrEmbed-5-6

Export a STEP AP214 file from another CAD system

* Create or open an assembly in another CAD system, for example [SolidWorks](https://www.solidworks.com/) or the free, open-source application [FreeCAD](https://www.freecadweb.org/).
* Export the assembly to a STEP AP214 file, ensuring the file extension is .STP or .STEP or the lower-case equivalent.

Read a STEP file using StrEmbed-5-6

* Run StrEmbed-5-6 by any of the methods described in the previous section.
* Open a file via the “File open” tool and then select a STEP or STP file to open, as shown in Figure 1. The example used hereafter for illustration is “Torch Assembly.STEP”, which consists of a simple torch of ten distinct parts and is bundled with StrEmbed-5-6. Some other examples are also presented.

|  |  |
| --- | --- |
|  | C:\Users\prehr\.spyder-py3\_GUI development\StrEmbed-5-3\_5-3 users manual and notes\fileopen3.png |
| Figure 1: Opening a STEP or STP file. Left: via tool in “Home” tab. Right: file dialog. | |

* After a STEP file is loaded, the corresponding assembly tree structure is displayed in the “parts view” on the left and as a Hasse diagram (a kind of directed graph visualisation) in the “lattice view” on the right. Images of parts and assemblies are displayed as toggle buttons in the “selector view” in the top-centre when the corresponding check boxes in the parts view are activated. The “3D view” in the bottom-centre shows (a) all parts selected in any of the other views, or (b) all parts in the assembly, if none is selected elsewhere. The user interface with the torch example loaded is shown in Figure 2.
* Additional functionality is provided in the lattice view, allowing the user to control which part of the network is visible, and to save images, for example, *via* [Matplotlib](https://matplotlib.org/)’s [FigureCanvas](https://matplotlib.org/tutorials/intermediate/artists.html). In the 3D view, the user can modify the view of parts with the mouse as follows:
  + Left button: hold to rotate
  + Middle button: hold to move
  + Right button: hold to zoom
* Parts and assemblies can be selected in the parts, selector and lattice views: in the parts view by single left mouse click (and multiple items can be selected by holding the SHIFT or CTRL keys); in the selector view by single left click (which toggles the selected item); and in the lattice view by right or left single click. Selections in one view are reflected immediately in the others.
* Please note that redundant subassemblies (*i.e.* any with only one child) are removed from the assembly structure upon loading from a STEP file, or upon modification of the assembly structure (see next section) by the user *via* StrEmbed-5-6’s interface, *e.g.* deleting a node and thereby leaving a subassembly with a single child. This is to ensure operations on the assembly adhere to rules dictated by lattice theory.
* When interacting with the lattice view, if the user clicks the mouse away from any existing nodes, a new assembly will be created (in green) consisting of two new items: one is a subassembly at the click position and another is a part or subassembly containing all parts not contained by the first. This functionality is intended to demonstrate the power of lexicographic ranking/unranking, which allows very large assemblies to be visualised and manipulated without realisation of all possible combinations of parts.

Modifying assembly structure

* A variety of operations can be performed on any parts and assemblies that are selected, either via the toolbar icons or via pop-up menu activated by right-clicking on selected items in the parts view. These operations are described in Table 1.

| Table 1*:* Toolbar operations. | | |
| --- | --- | --- |
| *Operation* | *Description* | *Toolbar icon* |
| *“Home” tab* | | |
| Assemble | Construct a new sub-assembly of all selected items | C:\Users\prehr\.spyder-py3\_GUI development\Images\assemble.bmp |
| Flatten | Remove all sub-assemblies from within the selected item, creating a flat list | C:\Users\prehr\.spyder-py3\_GUI development\Images\flatten.bmp |
| Disaggregate | Create assembly from single selected part | C:\Users\prehr\.spyder-py3\_GUI development\Images\disaggregate.bmp |
| Aggregate | Reduce sub-assembly to single item by removing all children; child IDs are retained for later use | C:\Users\prehr\.spyder-py3\_GUI development\Images\aggregate.bmp |
| Add node | Add new item to a selected sub-assembly | C:\Users\prehr\.spyder-py3\_GUI development\Images\add_node.bmp |
| Remove node(s) | Remove selected item(s) | C:\Users\prehr\.spyder-py3\_GUI development\Images\remove_node.bmp |
| Toggle sort type | Toggle between sorting child items by unique ID (default) or alphabetically according to description | C:\Users\prehr\.spyder-py3\_GUI development\Images\sort_mode.bmp |
| Reverse sort | Toggle between forward (default) and backward sort order; does not change sort type | C:\Users\prehr\.spyder-py3\_GUI development\Images\sort_reverse.bmp |
| *“Assistant” tab* | | |
| Map assembly elements | Create associations between parts in two assemblies based on pair-wise similarity, and seek user input where associations are not clear. (Not yet implemented.) | C:\_Work\_DCS project\__ALL CODE\_Repos\StrEmbed-5-5\StrEmbed-5-5 for git\Images\injection.png |
| Calculate similarity | Calculate the similarity, which is a user-defined aggregate of part data, assembly structure and part geometry, between two assemblies. (Not yet implemented.) | C:\_Work\_DCS project\__ALL CODE\_Repos\StrEmbed-5-5\StrEmbed-5-5 for git\Images\compare.png |
| Reconcile assemblies | Calculate minimum number of operations necessary to transform one assembly into another using node/edge additions and deletions; report transformation cost to user (see next section) | C:\Users\prehr\.spyder-py3\_GUI development\StrEmbed-5-3\Images\tree_small.bmp |
| Suggest new assembly | Create new assembly based on user preferences and machine learning technology. (Not yet implemented.) | C:\_Work\_DCS project\__ALL CODE\_Repos\StrEmbed-5-5\StrEmbed-5-5 for git\Images\bulb_sharp.png |

* Figure 3 and Figure 4 show examples of assembly operations performed via pop-up menus, before and after the operations. The changes are reflected immediately in the lattice view.

|  |  |
| --- | --- |
|  |  |
| Figure 3: “Assemble” operation via right-click pop-up menu. Left: before. Right: after. | |

|  |  |
| --- | --- |
|  |  |
| Figure 4: “Aggregate” operation via right-click pop-up menu. Left: before. Right: after. | |

* In addition to the toolbar/pop-up menu functionality described above, items in the parts view can be moved arbitrarily by dragging and dropping using the left mouse button, either within the same sub-assembly (which causes the item and its siblings to be reordered as the user wishes) or to another sub-assembly, (which reparents the dragged item).
* Lastly, item descriptions can be modified by slow double left-clicking on the item text in the parts view.

Performing assembly reconciliation

* Clicking the assembly reconciliation button in the Assistant tab causes the software to calculate the minimum number of operations – node/edge additions and deletions – necessary to transform one assembly into another. Specifically, the current functionality allows an assembly from a STEP file to be transformed into an alternative assembly that is created (in green) when the user clicks the Hasse diagram. The result of the reconciliation operation is a report to the user of a list of all necessary operations, as shown in Figure 5.

|  |
| --- |
|  |
| Figure 5: Assembly reconciliation, with report. |

* The other tools in the Assistant tab are not yet fully implemented.

People

The DCS project is hosted by the University of Leeds. Its members are Alison McKay, David Hogg, Alan de Pennington, Mark Robinson, Tiziana Callari, Hau Hing Chau, Tom Hazlehurst and Hugh Rice.

How to track and report bugs or get help

Known bugs and issues are tracked at the Github repository [here](https://github.com/paddy-r/StrEmbed-5-6). Please log bug/issue reports, request help or give feedback there or via the e-mail address below.

Hugh Rice

School of Mechanical Engineering

University of Leeds

Leeds LS2 9JT

[h.p.rice@leeds.ac.uk](mailto:h.p.rice@leeds.ac.uk)