Pasting operations

Current state:
A string of text$_S$ stemming from some source paragraph S is pasted into some target paragraph T. Text$_S$ then brings all of its formatting with it, viz. both the text modifications applied selectively to text$_S$ itself or to substrings of it and the formatting of the entire S. As a result of the pasting operation, T then not only contains text$_S$ with its own modifications; the entire T now assumes the paragraph formatting of S. This behavior of LO Writer is 1) theoretically misguided and 2) impractical for the user:

1) The design does not distinguish properly between block-level elements and inline elements, to use HTML terminology. Block-level elements like a paragraph, a list item, a table cell etc. have their own formatting properties; and likewise inline elements like words or strings inside a block-level element have their own formatting properties. The two sets of properties are mutually independent both in theory and, to a large extent, in practice. S and T are two block-level elements which may differ in their formatting. For instance, S may be an enumeration item, while T may be an indented paragraph. The pasting operation converts T into an enumeration item.

One cannot counter this argument by saying that LO design does not obey the principles of HTML design. Terminology aside, the difference between the two levels is valid for all of text processing. It is observed, *i.a.*, by MS Word and, in fact, by any other decent text processor except LO/OO. The inference 'If paragraphs S and T contain an identical piece of subtext, then S and T must share their block-level formatting' is so obviously invalid that one wonders how LO developers can have been clinging to it for decades now.

2) This behavior is practically never what the user intends. Normally, the user wants to preserve the inline-formatting of text$_S$. If not, he can use 'Paste Special – Unformatted text'. A situation where the user intends to apply, simultaneously, the block-level formatting of S to T is hard to imagine. Most of the time, the user has to undo the effect of this mistaken design.

One might reply to this that if the user does not want to carry the source formatting over to T, he can just use 'Paste Special – Unformatted text'. However, this does not solve it, because this operation also strips text$_S$ of its entire inline-formatting. Moreover, if text$_S$ contains some field code, this gets lost by this pasting operation. As a consequence, there is in LO no way to paste a string into some target without disfiguring either the source string or the target.

I have marked this as a bug rather than an enhancement because it is a serious design mistake, obstructing the user's workflow.

Suggestion:
The problem can, in principle, be solved either with the Copy/Cut or with the Paste operation:
a) Custom Copy:

- Distinguish between a Default Copy (as before) and a Custom Copy.
- In Default Copy, a string of text is copied preserving its inline-formatting while leaving behind the formatting of its source block.
- In Custom Copy, the user has the following options:
    - Do not preserve any formatting.
    - Preserve only inline formatting (as in the Default).
    - Preserve both inline and block-level formatting.

b) Custom Paste

- Distinguish between a Default Pasting and a Special (or Custom) Pasting as before.
- In Default Pasting, a string of text is pasted into the target preserving its inline-formatting while leaving behind the formatting of its source block.
- In Custom Pasting, the user has the following options:
    - Do not preserve any formatting (as currently in 'Unformatted text').
    - Preserve only inline formatting (as in the new Default).
    - Preserve both inline and block-level formatting (as in the current default).

There may be arguments for either solution. Implementing both would not make much sense, since the option chosen at the moment of copying limits the pasting options.