

knowledge elements **106** may contain a reference to the corresponding re-usable software component **104**.

[0026] Referring now to FIG. 2, additional details will be provided regarding the form and contents of the knowledge elements **106** in one implementation. As discussed above with reference to FIG. 1, the knowledge elements **106** comprise data that identifies the capabilities and affinities of the re-usable software component **104** along with the mechanisms for integrating the re-usable software component **104** with other software components. The knowledge elements **106** may be specified using XML, another markup language, or another data model altogether.

[0027] In one implementation, the capabilities of the re-usable software component **104** are specified using capability metadata **202**. The capability metadata **202** specifies the capabilities of the re-usable software component. The capability metadata **202** may further describe the purpose of the re-usable software component **104** and how the re-usable software component **104** is relevant to the business or higher purpose of a re-user.

[0028] In the implementation shown in FIG. 2, the knowledge elements **106** also include affinity metadata **204** that describes the other software components with which the re-usable software component **104** may be integrated. In this way, the affinity metadata **204** describes how the re-usable software component **104** has affinity, or composition value, to other software components published by the same or another publisher. The affinity metadata **204** may also identify a data entity that the re-usable software component **104** operates on or requires to be in memory. As discussed briefly above, the capabilities identified by the capability metadata **202** may be scoped to particular software components identified by the affinity metadata **204**.

[0029] As shown in FIG. 2, the knowledge elements **106** may also include derivation metadata **206**. The derivation metadata **206** indicates whether data entities specified by the affinity metadata **204** may be inherited or specialized from another data entity. The derivation metadata **206** may also indicate whether any of the identified data entities is an equivalent of, or replacement for, another externally described data entity. According to embodiments, the derivation metadata **206** further indicates whether a union operation or a merge operation is to be performed upon any of the data entities and an externally described data entity for any of the identified capabilities.

[0030] According to embodiments, the knowledge elements **106** further include transformation metadata **208**. The transformation metadata **208** identifies the mechanisms for integrating the re-usable software component **104** with other software components. As discussed briefly above, for instance, the transformation metadata **208** may specify how the re-usable software component **104** is to be transformed to allow it to integrate with other software components, such as the application **102**. For instance, the knowledge elements **106** may specify transformations for performing the merge and union operations with an externally described data entity as discussed above. As also discussed above, the transformations identified by the transformation metadata **208** may be scoped to capabilities of the re-usable software component **104** identified by the capability metadata **202**.

[0031] According to other embodiments, the knowledge elements **106** also include re-user metadata **210**. The re-user metadata **210** is metadata specified by a re-user for use with the re-usable software component **104** and is typically speci-

fied after the initial publication of the re-usable software component **104**. For instance, in one implementation, a mechanism is provided that allows a re-user of the re-usable software component **104** to further specify the capability metadata **202**, the affinity metadata **204**, the derivation metadata **206**, and the transformation metadata **208** originally published with the re-usable software component **104**. In this way, a re-user can supplement or replace the metadata originally published with the re-usable software component **104** at a later time.

[0032] In another implementation, the re-user metadata **210** indicates whether the transformation metadata **208** was useful for an actual integration between the re-usable software component **104** and another software component. In this manner, a re-user of the re-usable software component can provide feedback within the knowledge elements **106** regarding the usefulness of the transformation metadata **208**. Other re-users may utilize this data in determining whether to utilize the re-usable software component **104**.

[0033] In yet another implementation, the re-user metadata **210** includes a popularity value for the re-usable software component **104**. The popularity value indicates the popularity of the re-usable software component **104**. Other re-users may also utilize the popularity value in determining whether to utilize the re-usable software component **104**. It should be appreciated that a re-user of the re-usable software component **104** may also add other types of metadata to the re-user metadata **210** through an appropriate mechanism to provide feedback regarding the usefulness of the re-usable software component **104**. Additional details regarding the use of the knowledge elements **106** for discovering and integrating the re-usable software component **104** will be provided below with respect to FIGS. 3 and 4.

[0034] Turning now to FIG. 3, additional details will be provided regarding the embodiments presented herein for re-using software components. In particular, FIG. 3 is a flow diagram illustrating one method for publishing the re-usable software component **104**. It should be appreciated that the logical operations described herein are implemented (1) as a sequence of computer implemented acts or program modules running on a computing system and/or (2) as interconnected machine logic circuits or circuit modules within the computing system. The implementation is a matter of choice dependent on the performance and other requirements of the computing system. Accordingly, the logical operations described herein are referred to variously as states operations, structural devices, acts, or modules. These operations, structural devices, acts and modules may be implemented in software, in firmware, in special purpose digital logic, and any combination thereof. It should also be appreciated that more or fewer operations may be performed than shown in the figures and described herein. These operations may also be performed in a different order than those described herein.

[0035] The routine **300** begins at operation **302**, where the capability metadata **202** is specified for the re-usable software component **104**. From operation **302**, the routine **300** proceeds to operation **304**, where the affinity metadata **204** is specified for the re-usable software component **104**. The routine **300** then continues to operation **306**, where the derivation metadata **206** is specified. From operation **306**, the routine **300** continues to operation **308**, where the transformation metadata **208** is specified.

[0036] After the capability metadata **202**, affinity metadata **204**, derivation metadata **206**, and transformation metadata