I can think of three things that need tracking :

* An item is physically located inside another item which can contain any number of items of any time (eg. a cupboard)

* An item is plugged into a slot which is part of a parent item. The parent has a number of slots of various types. Items can only plug into slots of the appropriate type.

* An item fulfills a role in a system. For example, a system built around a laptop might need a wireless network interface. This could be either a USB WLAN card or a mini-PCIe card.

I think the first two types are orthogonal: a USB stick is either in a cupboard, or plugged into a laptop (which might be in the cupboard). But I think it would be wrong to allow the item-plugged-into-a-slot to be in a different location from the item-with-the-slot.

One way to handle the location in a container would be to make a 'slot' able to hold more than one item, and an item to be able to be installed into more than one slot at a time. This seems like an abuse of the system.

I am not very happy with the 'location' modelling. There needs to be some more subtypes, so that an item has either a location or is installed in a slot; these should be mutually exclusive. But we can deal with this in the code.

The 'role' is slightly different. A remote-monitoring system might need a 'server' somewhere and 'sensors' somewhere else. But in other respects, an item playing a role in a system is very similar to a slot. I think this belongs to completely different domain.

Another aspect to think about is data generated from devices in a system. This is definitely outside the scope of this 'inventory management' problem.

Inventory

Containment_history
* uid: container_history_uid_t
+ item: item_uid_t {R3}
+ container_item: item_uid_t
+ timestamp: datetime_t
+ action: containment_action_e

Containment actions
* added_to_container
* removed_from_container
* add_container_full
* add_already_in_container
* remove_not_in_container

Installation_history
* uid: installation_history_uid_t
+ item: item_uid_t {R5}
+ slot: slot_uid_t {R5}
+ timestamp: datetime_t
+ action: installation_action_e

Installation actions
* installed_in_slot
* removed_from_slot
* install_already_in_slot
* install_slot_occupied
* remove_not_in_slot

Common static read-only items
* manufacturer: utf8_string
* model_number: utf8_string
* serial_number: utf8_string
* address: formatted_text
* location: utf8_string
* description: formatted_text

Other static read-only items
* modem IMEI: utf8_string
* SIM phone number: utf8_string
* roller diameter
* roller magnets
* hub WLAN SSID
* hub WLAN PSK

Example dynamic read-only items
* roller raw counter
* roller temperature
* roller battery level
* roller SNAP RSSI level
* roller uptime
* roller boot counter
* hub temperature
* hub SNAP RSSI level
* hub WLAN RSSI level
* hub modem RSSI level
* hub uptime
* hub boot counter
* server uptime
* server boot counter
* server job counter

R3
contains 0..*
is_contained_in 0..1

Item
* uid: item_uid_t
+ common_name: string_t
+ format_string: string_t

+ is_contained_in: item_uid_t {R3}
+ maximum_items_in_container: integer_t
+ containment_warning_threshold: integer_t
- _contains: item_uid_t {R3}

+ can_fit_into: slot_type_uid_t {R4}
- _is_installed_in: slot_uid_t {R5}
- _has_slots: slot_uid_t[ ] {R6}

+ is_specified_by: item_spec_uid_t {R11}

has_installed 0..1
R5
is_installed_in 0..1

Slot
* uid
+ slot_name: string_t
+ is_part_of_item {R6}
+ accepts_item_that_installs_in: slot_type_uid_t {R7}
+ has_installed: item_uid_t {R5}
+ is_specified_by_slot_spec: slot_spec_uid_t {R12}

has 0..*
is_part_of 1
R6

is_a_data_item_for 1
R21
has 0..*

specifies 0..*
describes_slot_it_fits_into 0..*

describes_item_it_accepts 1
is_specification_for 0..*

Slot_type
* uid: slot_type_uid_t
+ type_name: string

R4
can_fit_into 1
R7
accepts_item_that_installs_in 1

USB
miniUSB
micrUSB
PCI
PCIe
mPCIe
DDR
CPU

is_of_slot_type 1
R15

R12

Derived_data_value
* uid: data_value_uid_t
+ source_value: data_value_uid_t {R}

Raw_data_value
* uid: data_value_uid_t
+ data_item

has_raw_data_values 0..*
R22
is_a_data_value_for 1

Data_item
* uid: data_item_uid_t
+ name: string_t
+ data_item_: boolean_t
+ update_interval: duration_<x>_t
- _data_formats: data_format_uid_t [ ] {R23}

is_derived_value_of 0..*

is_a_data_format_for 0..*

R11

Data_value
* uid: data_value_uid_t
+ sample_time: timestamp_<x>_t
+ value
- derived_values: data_value_uid_t [ ] {R}

R12
is_source_value_for 1

speciifies_data_format_for 0..*
R12
is_formatted_as 1

Item_data_format_spec
* uid: data_format_spec_uid_t
+ name
+ description
+ data_item: data_item_uid_t {R23}
+ data_format: data_format_uid_t {R23}

R23

is_specified_by 0..1
describes_slot_type_for 0..*
is_specified_by 0..1

Item_spec
* uid: item_uid_t
+ manufacturer: string_t
+ model_number: string_t
+ variant: string_t
- _has_slots_specified_by: slot_spec_t [ ] {R14}
- _specifies: item_uid_t [ ] {R11}

specifies_slots_for 1
R14
has slots_specified_by 0..*

Slot_spec
* uid: slot_spec_uid_t
+ slot_name: string_t
+ is_of_slot_type: slot_type_uid_t {R15}
+ specifies_slot_for: item_spec_t {R14}
- _is_specification_for: slot_uid_t [ ] {R12}

has_data_formatted_as 0..*

Data_conversion
* uid: data_conversion_uid_t
+ name
+ forward_function
+ reverse_function

specifies_input_format_for 0..*
R12
has_input_format_specified_by 1

specifies_output_format_for 0..*
R12
has_output_format_specified_by 1

Data_format
* uid: data_format_spec_uid_t
+ name
+ description

Data formats
* utf8_string
* formatted_text
* bytes4_ltc2498_adc_raw
* int16le_temperature_C_10
* int16le_mass_kg_10
* b16_4_mass_kg

Data conversion

Data values are either configuration values (writeable) or data value (read-only).

Configuration values might change the operation of the system or writeable. A writeable value can be changed from 'the outside' and

A non-updating read-only
An updating data value changes periodically. The device generates a 'raw' value in a particular format. This value need to be changed to a number of different formats through a chain of conversion functions; this chain of data values is different for each data item.

Inventory

A container has items stored in it -- this is described by association R1. Note that a container can be empty, but it is still a container that just happens to have nothing in it. This is shown by hte '0..*' multiplicity on one end of the relationship.

Each item is contained in exactly 1 container. This is shown by the '1' multiplicity on the other end of R1.

The relationship is formalised by the attrtibute 'container' in the Item. This is effectively a pointer to its container.

In this abstract model, the container does not know what it contains. To find that out, you have to search through all items to find those with the matching 'container'. However, in a real implementation, the container might have a list of its contents; this will add to the complexity of the data storage, but will usually improve performance.

Each item in the container can be either another container or a leaf-item -- this is shown by the disjoint-extends relationship R2.

Simultaneously, each item is either a 'room', a 'cabinet' or a 'device'. The difference between them is the descriptive data that each one has. A room has a physical address and a lat/long location. A cabinet has a physical description (eg. the grey cabinet with scratches) and a relative location ('under the window'). A device is described by the manufacturer, model number and serial number.

There might also be other classes of item.

This model allows a device to contains other devices (eg. a laptop might have a graphics card).

It also allows a device (eg. a laptop) to contain a room. You just have to look after your database to prevent that.

To duplicate an item with no item_spec

- parameters: serial_number
- copy manufacturer, model and variant from source
- copy slots, but not slot occupancy data

To duplicate an item with an item_spec, copy the data from the item_spec and slot_specs

To duplicate an entire subassembly, you really do need an item_spec for each item in the subasssm

- parameters: serial_number for each item in the new subassembly
- duplicate top item
- duplicate each sub-item, and link new the sub-item into the corresponding slot (from the slot_spec) in the parent-item
- link the